



UNIVERSIDAD
NACIONAL
DE LOJA



Área de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Ingeniería en Sistemas

"DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES Y/O CONGESTIONAMIENTO VEHICULAR."

*"Tesis previa a la obtención del título de
Ingeniero en Sistemas".*

Autores:

- Andrés Armando Armijos Armijos
- Pedro Fernando Aponte Rueda

Director: Ing. Paz Arias Henry Patricio, Mg. Sc

Loja-Ecuador

2015

CERTIFICACIÓN DEL DIRECTOR

Mg. Sc. Henry Patricio Paz Arias.

DOCENTE DE LA CARRERA DE INGENIERÍA EN SISTEMAS, DEL ÁREA DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES DE LA UNIVERSIDAD NACIONAL DE LOJA.

CERTIFICA:

Haber asesorado y revisado detenida y minuciosamente durante todo su desarrollo, el Proyecto de Fin de Carrera titulado: “**DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES Y/O CONGESTIONAMIENTOS VEHICULAR**”. Realizado por los postulantes Andrés Armando Armijos Armijos y Pedro Fernando Aponte Rueda.

Por lo tanto, autorizo proseguir los trámites legales pertinentes para su presentación y defensa.

Martes 10 de Marzo de 2015.



Mg.Sc. Henry Patricio Paz Arias.
DIRECTOR DEL PFC.

AUTORÍA

Nosotros **ANDRÉS ARMANDO ARMIJOS ARMIJOS Y PEDRO FERNANDO APONTE RUEDA** declaramos ser autores del presente trabajo de tesis y eximimos expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente aceptamos y autorizamos a la Universidad Nacional de Loja, la publicación de nuestra tesis en el repositorio institucional – Biblioteca Virtual.

Firma: 

Cédula: 1104418098

Fecha: 31 de Julio del 2015

Firma: 

Cédula: 1103983498

Fecha: 31 de Julio del 2015

CARTA DE AUTORIZACION

Nosotros **ANDRÉS ARMANDO ARMIJOS ARMIJOS Y PEDRO FERNANDO APONTE RUEDA**, declaramos ser autores de la tesis titulada: **DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES Y/O CONGESTIONAMIENTO VEHICULAR**, como requisito para optar al grado de: **INGENIERO EN SISTEMAS**; autorizamos al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los 31 días del mes de Julio del dos mil quince.

Firma: 

Autor: Andrés Armando Armijos Armijos.

Cédula: 1104418098

Dirección: Cdla. Daniel Álvarez.

Teléfono: 073026853 **Celular:** 0990898699

Correo: adrearmi@gmail.com

Firma: 

Autor: Pedro Fernando Aponte Rueda.

Cedula: 1103983498

Dirección: Monte de los Olivos.

Teléfono: 072540352 **Celular:** 0991419277

Correo: fernandoaponte1103@gmail.com

DATOS COMPLEMENTARIOS

Director de tesis: Ing. Henry Patricio Paz Arias, Mg.Sc.

Tribunal de grado: Ing. Luis Roberto Jácome Galarza, Mg.Sc.

Ing. Ana Lucia Colala Troya, Mg.Sc.

Ing. Alex Vinicio Padilla Encalada, Mg.Sc.

DEDICATORIA

Andrés Armijos

Primeramente a Dios por darme la vida, a mis Padres, a mis hermanos, familiares y amigos quienes han estado conmigo en los buenos y malos momentos de forma incondicional para con ello lograr culminar mi Carrera.

Pedro Aponte

Agradezco a Dios por todas las bendiciones y por permitirme culminar una de mis grandes metas. Así mismo le doy gracias a mis Padres y hermanos que siempre estuvieron apoyándome incondicionalmente cuando más los necesite.

AGRADECIMIENTO

El presente trabajo de tesis primeramente nos gustaría agradecer a DIOS, por bendecirnos, por permitirnos llegar hasta aquí, porque hiciste realidad este sueño anhelado.

A la UNIVERSIDAD NACIONAL DE LOJA en especial al ÁREA DE LA ENERGÍA LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES por darnos la oportunidad de estudiar y ser profesionales.

A la Carrera de Ingeniería en Sistemas; al personal de sus dignas Autoridades y Docentes que desinteresadamente imparten sus conocimientos, los que nos sirvieron de guía en la formación académica profesional.

A nuestro director de tesis Ing. Henry Paz por su esfuerzo y dedicación, quien con sus conocimientos, su experiencia, su paciencia y su motivación ha logrado que terminemos nuestros estudios con éxito.

A nuestros Padres, familiares, compañeros, amigos quienes nos han apoyado de forma desinteresada para que este día llegue a ser una realidad.

A todos, nuestra eterna gratitud. Y de todo corazón que Dios los bendiga.

LOS AUTORES

Índice de Contenidos

Índice General

CERTIFICACIÓN DEL DIRECTOR	II
AUTORÍA	III
CARTA DE AUTORIZACION	IV
DEDICATORIA.....	V
AGRADECIMIENTO	VI
Índice de Contenidos	VII
Índice General	VII
Índice de figuras	XI
Índice de tablas	XIII
Índice de diagramas	XV
a. Título	1
b. Resumen.....	2
Summary	3
c. Introducción	4
d. Revisión de Literatura.....	6
1. Capítulo 1: Visión artificial.....	6
1.1. Antecedentes.....	6
1.2. Introducción a la visión artificial.....	8
1.2.1. Definición de Visión.....	9
1.2.2. Definición de visión artificial.....	9
1.3. Etapas de la visión artificial	9
1.4. Áreas de aplicación	12
1.5. Ejemplos de Visión Artificial	13
2. Capítulo 2: Procesamiento digital de imágenes	15
2.1. Píxel.	15
2.2. Imagen digital.....	16
2.3. Clasificación de imágenes digitales	18

2.4.	Formatos de imágenes.....	21
2.5.	Filtros más utilizados en el procesamiento de imágenes.	24
2.5.1.	Bordes de Sobel.....	24
2.5.2.	Bordes de Canny.	24
2.5.3.	Bordes de Prewitt.....	25
2.5.4.	Bordes de Roberts.	26
2.6.	Técnicas para el procesamiento de imágenes.	27
2.6.1.	Escala de grises.....	27
2.6.2.	Binarización.....	27
2.6.3.	Umbralización.	28
2.6.4.	Redes neuronales	29
2.6.5.	Viola Jones.....	31
2.6.5.1.	Haar-like features	31
2.6.5.2.	Imagen Integral.....	33
2.6.5.3.	El clasificador	34
2.6.5.4.	Clasificador simple	35
2.6.5.5.	Algoritmo Adaboost.....	35
2.6.5.6.	El detector de clasificadores en cascada	38
2.6.5.7.	Entrenamiento de la cascada.....	39
3.	Capítulo 3. Herramientas, técnicas y librerías utilizadas en el desarrollo del sistema. 41	
3.1.	Herramientas	41
3.1.1	Cámara.....	41
3.1.1.1.	Cámara digital	41
3.1.1.2.	Cámara ip	41
3.1.1.3.	Cámara usb	41
3.1.1.4.	Cámara integrada en celular	41
3.1.2.	Ubuntu	42
3.1.3.	Lenguaje C++.....	43
3.1.4.	Qt.....	43
3.2.	Técnicas.....	44
3.2.1.	Descriptor de características	44

3.2.2.	Clasificadores.....	44
3.2.3.	Detección de contornos.	44
3.3.	Librerías	45
3.3.1.	Opencv	45
3.3.2.	CuteReports	46
3.3.3.	Librerías multimedia.....	46
e.	Materiales y Métodos.....	47
1.	Materiales	47
1.1.	Talento Humano	47
1.2.	Servicios.....	47
1.3.	Recursos Hardware y Software.....	48
1.4.	Materiales de Oficina	49
2.	Métodos.....	50
2.1.	Método Deductivo.....	50
2.2.	Método Inductivo	50
2.3.	Modelo en cascada.....	50
3.	Técnicas	52
f.	Resultados	53
1.	Análisis de Requerimientos	53
1.1.	Requerimientos Funcionales.....	53
1.2.	Requerimientos no Funcionales.....	54
2.	Diseño	55
2.1.	Algoritmo para la detección de vehículos y posible congestionamiento	55
2.2.	Algoritmo para la detección de vehículos y posible accidente	56
2.3.	Prototipado.....	57
3.	Implementación.....	60
3.1.	Construcción del clasificador en cascada	60
3.2.	Clasificador en cascada para la detección de vehículos	60
3.3.	Implementación de los algoritmos.....	60
3.4.	Codificación del algoritmo para la detección de vehículos, posible congestionamiento y posible accidente.	60
3.4.1.	Librerías y cabeceras.....	60

3.4.2.	Función para ingresar los puntos a través del Ratón (Mouse).	61
3.4.3.	Área de interés delimitada	62
3.4.4.	Implementación del clasificador en cascada	63
3.4.5.	Temporizador para congestionamiento	64
3.4.6.	Temporizador para accidente	65
3.4.7.	Comparador de desplazamiento o distancia	65
3.4.8.	Función para insertar los eventos en la base de datos.	66
4.	Pruebas	67
4.1.	Detección de vehículos	67
4.1.1.	Pruebas con imágenes	67
4.1.2.	Pruebas con video (cámara ip)	71
4.2.	Detección de personas	74
4.2.1.	Pruebas con imágenes	74
4.2.2.	Pruebas con video (cámara IP)	77
4.3.	Detección de vehículos y personas.	79
4.3.1.	Pruebas con imágenes	79
4.3.2.	Pruebas con video (cámara IP)	81
4.4.	Registro del evento de posible congestionamiento	85
4.5.	Registro del evento de posible accidente	87
g.	Discusión.	89
h.	Conclusiones	91
i.	Recomendaciones	92
j.	Bibliografía	94
k.	Anexos	98

Índice de figuras

Figura 1. Test de Turing.....	8
Figura 2. Etapas de la visión artificial	10
Figura 3. Etapas de la V.A. por niveles	12
Figura 4. Imagen original	15
Figura 5. Zoom de la imagen [9].....	16
Figura 6. Representación imagen digital	16
Figura 7. Imagen monocroma codificada en 256 niveles de intensidad (niveles de gris) ..	17
Figura 8. La línea amarilla en el histograma indica el valor medio de intensidad.	18
Figura 9. Binarización de una imagen	19
Figura 10. Imagen en escala de grises	20
Figura 11. Dos formatos de imágenes digitales.	21
Figura 12. Ampliación de imágenes.	22
Figura 13. Imagen original e imagen con filtro de sobel	24
Figura 14. Imagen original e imagen con filtro de Canny	25
Figura 15. Imagen original e imagen con filtro de prewitt.	26
Figura 16. Imagen original e imagen con filtro de Roberts.	26
Figura 17. Imagen en modo escala de grises	27
Figura 18. Función de transformación y operador intervalo de umbral binario	28
Figura 19. Imagen original e imagen binarizada	28
Figura 20. Función de transformación y grafica del operador umbral.	29
Figura 21. Imagen original y segmentación con umbral de 128	29
Figura 22. Modelo de una neurona artificial	30
Figura 23. Características de 2 rectángulos.....	32
Figura 24. Características de 3 y 4 rectángulos.....	32
Figura 25. Representación de una imagen genérica con un modelo de característica.	33
Figura 26. Valor de la imagen integral en un punto (x, y).	33
Figura 27. Resultado de la imagen integral.....	34
Figura 28. Descriptores rectángulos.....	37
Figura 29. Estructura de clasificadores en cascada	38
Figura 30. Ubuntu. Última versión disponible 14.04 LTS.....	42
Figura 31. Qt, Última versión 4.8.....	43
Figura 32. OpenCV, última versión 3.0	45
Figura 33. Metodología de Desarrollo del ciclo de vida clásico	51
Figura 34. Pantalla principal del sistema.....	57
Figura 35. Pantalla detección posibles accidentes	57
Figura 36. Pantalla detección posibles congestionamientos	58
Figura 37. Pantalla generación de reportes de eventos de accidentes.....	58
Figura 38. Pantalla generación de reportes de eventos de congestionamientos.....	58
Figura 39. Pantalla generación de reportes de eventos de congestionamientos.....	59
Figura 40. Pantalla Acerca del programa	59

Figura 41. Área de interés trazada	63
Figura 42. Ejemplo detección en lluvia.....	67
Figura 43. Ejemplo de detección en llovizna con sol	68
Figura 44. Ejemplo de detección en día con sol	69
Figura 45. Ejemplo de detección de vehículos en día sombreado	70
Figura 46. Detección a través de cámara ip, Terminal Terrestre	71
Figura 47. Detección a través de cámara ip.....	72
Figura 48. Detección a través de cámara ip.....	73
Figura 49. Ejemplo de detección de personas en calle Mercadillo.....	74
Figura 50. Ejemplo de detección de personas en estacionamiento	75
Figura 51. Ejemplo de detección de personas con imágenes de internet.....	76
Figura 52. Detección de personas calle Mercadillo	77
Figura 53. Detección de personas en estacionamiento Alameda	78
Figura 54. Detección de personas y vehículos paso peatonal calle Mercadillo	79
Figura 55. Detección de personas y vehículos estacionamiento Alameda Real.....	80
Figura 56. Detección de personas y vehículos calle Mercadillo	81
Figura 57. Detección de personas y vehículos estacionamiento Alameda	82
Figura 58. Detección de posible congestionamiento.	86
Figura 59. Registro del posible congestionamiento.	86
Figura 60. Detección de posible accidente.	88
Figura 61. Registro de posible accidente	88

Índice de tablas

TABLA 1. TRABAJOS DE INTELIGENCIA ARTIFICIAL	6
TABLA 2. EJEMPLOS DE VISIÓN ARTIFICIAL	13
TABLA 3. ALGORITMO DE ADABOOST.....	36
TABLA 4. TALENTO HUMANO.....	47
TABLA 5. SERVICIOS	48
TABLA 6. HARDWARE Y SOFTWARE	48
TABLA 7. MATERIALES DE OFICINA.....	49
TABLA 8. PRESUPUESTO FINAL	49
TABLA 9. REQUERIMIENTOS FUNCIONALES DEL SISTEMA	53
TABLA 10. REQUERIMIENTOS NO FUNCIONALES DEL SISTEMA.....	54
TABLA 11. LIBRERÍAS Y CABECERAS.....	60
TABLA 12. FUNCIÓN PARA INGRESAR LOS PUNTOS A TRAVÉS DEL RATÓN.	61
TABLA 13. FUNCIÓN PARA DELIMITAR EL ÁREA DE INTERÉS.	62
TABLA 14. FUNCIÓN DONDE SE IMPLEMENTA LOS CLASIFICADORES.....	63
TABLA 15. FUNCIÓN TEMPORIZADOR CONGESTIONAMIENTO.	64
TABLA 16. FUNCIÓN TEMPORIZADOR ACCIDENTE.....	65
TABLA 17. FUNCIÓN DISTANCIA O DESPLAZAMIENTO DE LOS VEHÍCULOS.....	66
TABLA 18. FUNCIÓN PARA INSERTAR LOS EVENTOS EN LA BASE DE DATOS.....	66
TABLA 19. DETECCIÓN VEHÍCULOS EN LLUVIA	67
TABLA 20. DETECCIÓN DE VEHÍCULOS CON LLOVIZNA Y SOL	68
TABLA 21. DETECCIÓN DE VEHÍCULOS EN DÍA CON SOL	69
TABLA 22. DETECCIÓN DE VEHÍCULOS EN DÍA SOMBREADO	70
TABLA 23. DETECCIÓN DE VEHÍCULOS EN LLUVIA CON CÁMARA IP	71
Tabla 24. DETECCIÓN VEHICULAR DÍA SOMBREADO, TERMINAL TERRESTRE	72
TABLA 25. DETECCIÓN DE VEHÍCULOS PASO PEATONAL MERCADILLO	73
TABLA 26. DETECCIÓN DE PERSONAS PASO PEATONAL CALLE MERCADILLO.....	74
TABLA 27. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL	75
TABLA 28. DETECCIÓN DE PERSONAS IMÁGENES INTERNET	76
TABLA 29. DETECCIÓN DE PERSONAS PASO PEATONAL CALLE MERCADILLO.....	77
TABLA 30. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL	78
TABLA 31. DETECCIÓN DE PERSONAS Y VEHÍCULOS CALLE MERCADILLO.....	79
TABLA 32. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL	80
TABLA 33. DETECCIÓN PERSONAS Y VEHÍCULOS CALLE MERCADILLO	81
TABLA 34. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL	82
TABLA 35. DETECCIÓN DE VEHÍCULOS CON IMÁGENES.....	83
TABLA 36. DETECCIÓN DE PERSONAS CON IMÁGENES	83
TABLA 37. DETECCIÓN DE VEHÍCULOS Y PERSONAS CON IMÁGENES.....	83
TABLA 38. DETECCIÓN DE VEHÍCULOS EN TIEMPO REAL	84
TABLA 39. DETECCIÓN DE PERSONAS EN TIEMPO REAL	84
TABLA 40. DETECCIÓN, VEHÍCULOS Y PERSONAS EN TIEMPO REAL.....	84

TABLA 41. PRUEBA CONGESTIONAMIENTO.....	85
TABLA 42. PRUEBA ACCIDENTE.....	87

Índice de diagramas

Diagrama 1. Diagrama de flujo del algoritmo para la detección de congestionamientos ..	55
Diagrama 2. Diagrama de flujo del algoritmo para la detección de accidentes	56
Diagrama 3. Vehículos detectados en lluvia	67
Diagrama 4. Detección de vehículos con llovizna y sol	68
Diagrama 5. Detección de vehículos en día con sol	69
Diagrama 6. Detección de vehículos en día sombreado	70
Diagrama 7. Detección de vehículos en lluvia con cámara ip	71
Diagrama 8. Detección de vehículos día sombreado, Terminal Terrestre	72
Diagrama 9. Detección de vehículos día sombreado, mercadillo sur-norte	73
Diagrama 10. Detección de personas paso peatonal calle Mercadillo	74
Diagrama 11. Detección de personas estacionamiento Alameda Real	75
Diagrama 12. Detección de personas imágenes internet	76
Diagrama 13. Detección de personas calle Mercadillo	77
Diagrama 14. Detección de personas estacionamiento Alameda	78
Diagrama 15. Detección de personas y vehículos paso peatonal calle Mercadillo	79
Diagrama 16. Detección de personas y vehículos estacionamiento Alameda Real	80
Diagrama 17. Detección de personas y vehículos paso peatonal calle Mercadillo	81
Diagrama 18. Detección de personas y vehículos estacionamiento Alameda Real	82
Diagrama 19. Prueba congestionamiento	86
Diagrama 20. Prueba accidente	88

a. Título

“DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES Y/O CONGESTIONAMIENTO VEHICULAR”.

b. Resumen

Actualmente el mundo está lleno de dispositivos capaces de realizar capturas de imágenes a grandes distancias, vigilar sectores específicos, cámaras de seguridad para las empresas, ojos de águila, etc. Todos estos dispositivos resultan de gran utilidad al momento de tomar una decisión por el ser humano, pero este no puede estar presente 24 horas, 365 días al año, trabajando y tomando decisiones sin tomarse un descanso, es ahí donde la visión artificial juega un papel fundamental, haciendo que los dispositivos mencionados se conviertan en dispositivos de capturas de imágenes pasivos a activos por sí solos, capaces de tomar una decisión como lo haría un ser humano pensante e inteligente, sin importar el horario.

En la actualidad, es necesario determinar el aforo vehicular de un determinado lugar donde con frecuencia existe congestiónamiento de vehículos, ya que el tránsito es controlado por semáforos con temporizadores que se sitúan en intersecciones viales para regular el normal tráfico vehicular y peatonal, pero resultan ineficientes al momento que se presentan aglomeraciones en horas pico, provocando así una serie de problemas de contaminación ambiental, accidentes de tránsito, contaminación por el ruido, etc.

Bajo este panorama, lo que se presenta es una propuesta enmarcada dentro del proyecto de semáforos inteligentes; dando a conocer ciertas herramientas y algoritmos que pueden ayudar a realizar la detección de posibles congestiónamientos y posibles accidentes de tránsito de una forma autónoma, sin la necesidad de que esté presente la autoridad competente.

El presente trabajo de titulación, comprende el análisis, diseño e implementación de un sistema basado en visión artificial para la detección de congestiónamientos y posibles accidentes de tránsito desde la perspectiva de un semáforo.

Summary

Currently the world is full of devices capable of to capture images over long distances, monitor specific sectors, security cameras for businesses, eagle eyes, etc. All these devices are very useful when making a decision by humans, but this cannot be present 24 hours, 365 days a year, working and making decisions without taking a break, this is where the machine vision plays a role key, making such devices become passive devices capture images of assets for themselves, able to make a decision as you would a thinking, intelligent human being, regardless of the schedule.

Nowadays, it is necessary to determine the vehicle capacity of a particular place where there is often congestion of vehicles and that traffic is controlled by traffic lights with timers that are at intersections to regulate the normal vehicular and pedestrian traffic, but are inefficient when you agglomerations peaking occur, causing a series of problems of environmental pollution, traffic accidents, noise pollution, etc.

Under this scenario, what is presented is a proposal framed within the project of intelligent traffic lights; revealing certain tools and algorithms that can help in the detection of possible congestion and traffic accidents autonomously, without the need for the competent authority is present.

This degree work includes analysis, design and implementation of a system based on artificial vision for detecting congestion and traffic accidents from the perspective of a traffic light system.

c. Introducción

El presente trabajo de titulación versa sobre un campo que en los últimos tiempos ha avanzado significativamente, ya que la evolución de la tecnología obliga a que los sistemas sean cada vez más autónomos, en otras palabras que no intervenga el ser humano, por tal razón se hace imprescindible en muchos sistemas dotarlos de alguna herramienta que describa su entorno o parte del mismo, debido a esa gran necesidad de automatizar ciertos sistemas y volverlos más precisos nace la Visión Artificial, que no es más que una rama o subcampo de la Inteligencia Artificial que tiene como propósito programar un computador para que entienda una escena o las características de una imagen. Tal vez en nuestro medio no se hayan visto muchos proyectos enfocados en esta línea de investigación, pero no por ello significa que no se lo pueda hacer, es más, eso es lo que la hace más interesante y atractiva; ya que se puede investigar, diseñar e implementar sistemas que solventen las necesidades tecnológicas de la población.

En muchas ciudades del mundo se están implementando sistemas que permiten regularizar y ordenar el tránsito terrestre, tal es el caso del foto peaje, que permite que el tránsito fluya sin la necesidad de que el vehículo se detenga, en otros países por no decir todos, se están utilizando este tipo de tecnologías para sancionar a los conductores cuando rebasan la velocidad permitida.

Como es de conocimiento general, las aglomeraciones y accidentes que ocurren dentro de las ciudades, provocan retrasos, desesperación, desorden y contaminación por ruido, por tal razón desarrollar sistemas computacionales que ayuden al mejoramiento del ordenamiento vehicular es de vital importancia.

Según las estadísticas es suficiente que existan 10 o más vehículos en un lapso de 60 a 120 segundos en una intersección para que se considere como un congestionamiento. Además la mayoría de accidentes se presentan en horas de la tarde y madrugada, casi por lo general entre dos vehículos.

El presente proyecto de tesis está estructurado de acuerdo a los lineamientos establecidos por la Universidad Nacional de Loja y el Área Energía, las Industrias y los Recursos Naturales No Renovables, primeramente se encuentra el resumen, que es una síntesis del

contenido del informe final del proyecto fin de carrera, luego tenemos la introducción, donde se describe de manera global el ámbito del proyecto y los objetivos que deben cumplirse.

En la etapa de análisis, interpretación y obtención de los requerimientos del presente trabajo se utilizaron varias técnicas de recolección de información y métodos científicos, que se aplicaron a las personas encargadas de la Agencia Nacional de Tránsito y personas que laboran en la unidad Municipal de Tránsito y Transporte Terrestre. Además por el tipo de sistema se optó por utilizar la observación directa como principal técnica. Así mismo en la etapa de diseño se estudiaron varias herramientas y algoritmos para determinar el mejor camino a seguir.

Para el desarrollo del sistema informático se optó por el lenguaje C++ a través del framework Qt bajo la plataforma Linux, sistema operativo Ubuntu 14.04. Y como gestor de base de datos se utilizó SQLite.

En la sección de resultados se encuentra detallado el proceso para el desarrollo del sistema siguiendo el modelo de ciclo de vida clásico, también denominado “modelo en cascada”, lo que permitió hacer un desarrollo rápido del sistema y poder enfocarse de manera directa en los requerimientos del mismo. En este contexto se describen los requerimientos del sistema, diseño de los algoritmos y de la misma forma se detalla la implementación, pruebas y el análisis de resultados.

En el apartado de discusión se explica cómo se alcanzaron cada uno de los objetivos planteados al inicio del proyecto. Luego se describe el apartado de conclusiones y recomendaciones que se realizaron en base al trabajo de titulación, y finalmente la bibliografía y anexos.

d. Revisión de Literatura

1. Capítulo 1: Visión artificial.

1.1. Antecedentes.

La psicología cognitiva ha adoptado la metáfora del ordenador para pensar sobre la mente. La mente procesa la información, la codifica, la almacena y la recupera como un ordenador. Nuestro cerebro es el hardware sobre el que corren programas que nos permiten hablar, ver o pensar (el software). La Inteligencia Artificial adopta la imagen especular y, en su versión fuerte, no de manera metafórica sino literal: Un ordenador es una mente. Los circuitos son distintos a los del cerebro y los programas con frecuencia también, aunque produzcan resultados semejantes a la conducta humana; pero cuando estos se ejecutan, la máquina piensa, igual que la mente cuando procesa la información.

La cuestión es, ¿puede una entidad con unos muy limitados sentidos artificiales, esto es, sin visión artificial, aunque tal vez con sensores de distancia o con percepción de contornos y formas pero sin reconocimiento semántico de las mismas; sin nariz electrónica, aunque con un tacto simple, sin apenas oído para localizar sonidos en el mejor de los casos; con efectores sin la flexibilidad ilimitada de las extremidades humanas, si posee alguno, con prótesis mecánicas, sin emociones, sin necesidades ni sentido común como son la mayoría de los ordenadores y robots, ser consciente del mundo, tener sentimientos, autoconciencia o teoría de la mente? [1].

TABLA 1. TRABAJOS DE INTELIGENCIA ARTIFICIAL

Año	Autores	Trabajo o invención
1943	Warren McCulloch y Walter Pitts	La IA comienza siendo computación neuronal con el trabajo teórico: “Un cálculo de las ideas inmanentes en la actividad nerviosa”, el cual hacía énfasis en la estructura física.
1949	Donald O. Hebb	Publica “La organización de la conducta”, que sirvió de base para los algoritmos de aprendizaje en las redes neuronales artificiales.

1950	Alan Turing	Se propone la prueba de Turing (o también «juego de imitación»), para examinar la inteligencia de una máquina. Alan Turing se puede considerar el padre de la Inteligencia Artificial (IA). Siendo el autor del concepto de computadora, predijo que la máquina podría llegar a adquirir una capacidad comparable con la inteligencia humana. La prueba se basa en la idea de que la interacción verbal constituye un medio en el que la inteligencia se hace más patente.
1956	John McCarthy	Acuña el término “Inteligencia Artificial” en la conferencia de Dartmouth, la primera conferencia dedicada a la IA.
1958	John McCarthy	Desarrolla el lenguaje LISP, lenguaje con el que se desarrollan la mayoría de sistemas expertos. Y el emacs (Richard Stallman).
1959	Newell, Shaw y Simon	Desarrollo del programa General Problem Solver (GPS).
1963	Edward A. Feigenbaum y Julian Feldman	Edward A. Feigenbaum y Julian Feldman Publicaron Computers and Thought, la primera colección de artículos de IA.
1968	Marvin Minsky y Simon Papert	Publican Perceptrons.
1972	Alain Colmerauer	Desarrolla el lenguaje PROLOG.
1975	Ted Shortliffe	Desarrollo del sistema de reglas de producción MYCIN en su tesis doctoral.
1986	Rumelhart, McClelland y el grupo PDP	Desarrollan el perceptrón multicapa y el algoritmo de aprendizaje por retropropagación del error (BP).
Actual	Google, Yahoo y Microsoft	En el tiempo contemporáneo empresas como Google, Yahoo y Microsoft invierten grandes cantidades de dinero a la investigación y desarrollo de sistemas cada vez más inteligentes. [2]

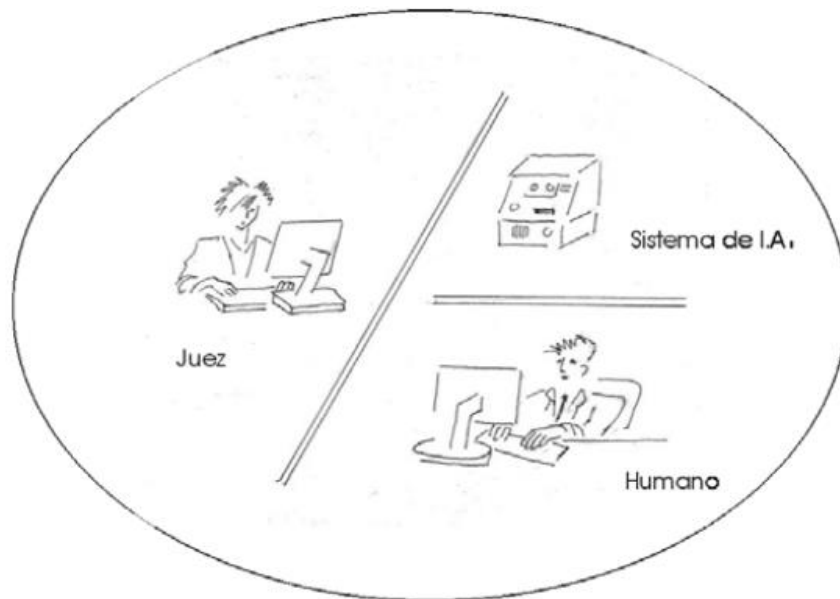


Figura 1. Test de Turing

En el test de Turing una persona (juez) ha de mantener una conversación (por medio de un interfaz y un teclado) con el sistema de IA y con un humano. Si el juez no puede diferenciar quién de los dos es humano con una probabilidad superior al azar, se puede considerar que el sistema posee una inteligencia comparable a la humana. [1]

1.2. Introducción a la visión artificial

Uno de los sentidos más importantes de los seres humanos es la visión. Ésta es empleada para obtener la información visual del entorno físico. Según Aristóteles, "Visión es saber que hay y donde mediante la vista".

De hecho, se calcula que más de 70% de las tareas del cerebro son empleadas en el análisis de la información visual. El refrán popular de "Una imagen vale más que mil palabras" tiene mucho que ver con los aspectos cognitivos de la especie humana. Casi todas las disciplinas científicas emplean utillajes gráficos para transmitir conocimiento. Por ejemplo, en Ingeniería Electrónica se emplean esquemas de circuitos, a modo gráfico, para describirlos. Se podría hacerlo mediante texto, pero para la especie humana resulta mucho

más eficiente procesar imágenes que procesar texto. La visión humana es el sentido más desarrollado y el que menos se conoce debido a su gran complejidad. Es una actividad inconsciente y difícil de saber cómo se produce. De hecho, hoy en día, se carece de una teoría que explique cómo los humanos perciben el exterior a través de la vista.

En el año 1826 el químico francés Niepce (1765-1833) llevó a cabo la primera fotografía, colocando una superficie fotosensible dentro de una cámara oscura para fijar la imagen. Posteriormente, en 1838 el químico francés Daguerre (1787-1851) hizo el primer proceso fotográfico práctico. Daguerre utilizó una placa fotográfica que era revelada con vapor de mercurio y fijada con trisulfato de sodio.

1.2.1. Definición de Visión

La visión es el sentido más importante que tiene el ser humano. Así, mientras, para el oído se tiene alrededor de treinta mil terminaciones nerviosas, en la vista hay más de dos millones. La radiación exterior recibida por el ojo debe ser transformada en señales que sean procesadas por el cerebro. El ojo es el elemento transductor mientras que el cerebro es el que procesa dicha información. [3]

1.2.2. Definición de visión artificial

Desarrollo de métodos y algoritmos que permitan comportarse a las computadoras de modo inteligente. [2]

1.3. Etapas de la visión artificial

Aunque cada aplicación de Visión Artificial tiene sus especificidades, se puede decir que existe un tronco común de etapas entre ellas. No necesariamente debe cubrirse todas en una implementación concreta. Hay algunas veces que sólo se tiene un subconjunto de las fases que se van a citar. Por otro lado, aunque la exposición muestra un encadenamiento temporal de una etapa sobre otra, no es real esta simplificación; se hace para facilitar la

comprensión y en la puesta en práctica siempre se encuentra realimentación entre las distintas fases.

La primera etapa es la construcción del sistema de formación de las imágenes. Su objetivo es realzar, mediante técnicas fotográficas (iluminación, óptica, cámaras, filtros, pantallas), las características visuales de los objetos (formas, texturas, colores, sombras). El éxito de muchas aplicaciones depende de un buen diseño en esta primera etapa.

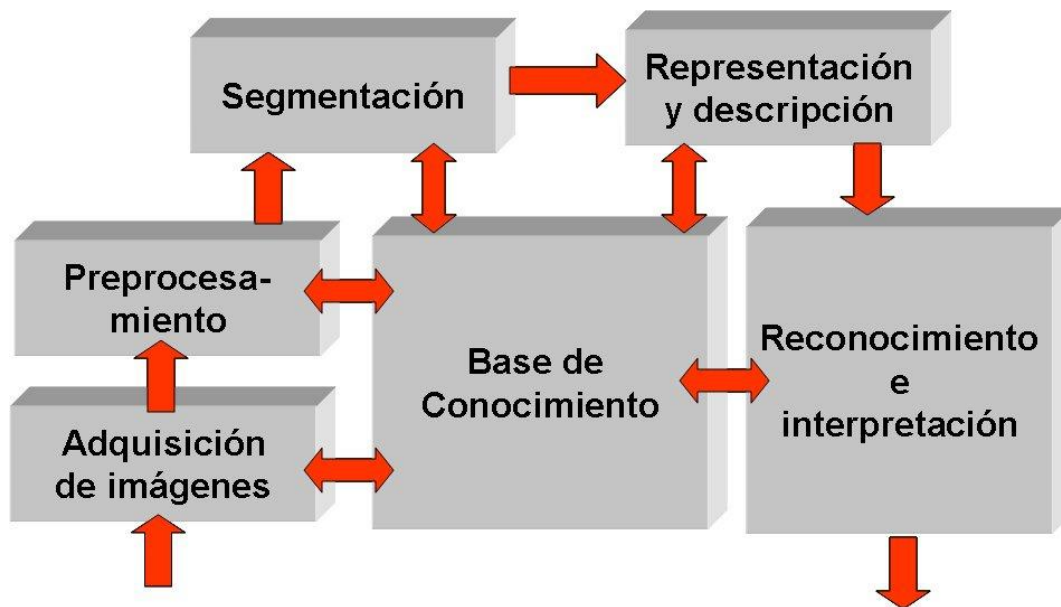


Figura 2. Etapas de la visión artificial

Una vez adquirida la imagen se pasará a la etapa de pre procesado. El objetivo es mejorar la calidad informativa de la imagen adquirida. Se incluyen operaciones de mejora de la relación señal-ruido, SNR, de atenuar las imperfecciones de la adquisición debido a la función de transferencia del sistema de captación de imágenes, de regularizar la imagen, de mejorar el contraste o de optimizar la distribución de la intensidad o de realzar algunas características de la imagen, como bordes o áreas.

Segmentación es la fase donde se particiona la imagen en áreas con significado.

Por ejemplo, en una imagen de satélite se determina las zonas de agua, de cultivo, urbanas, carreteras. Existen varias técnicas: umbralizaciones, discontinuidades, crecimiento de regiones, uso del color o de movimiento, etc. Estas estrategias serán analizadas en el capítulo quinto.

Una vez dividida la imagen en zonas con características de más alto nivel se pasará a su extracción de las características. Básicamente son de tipo morfológico, tales como área, perímetro, excentricidad, momentos de inercia, esqueletos, pero también se puede emplear características basadas en la textura o en el color.

Fíjese que se ha pasado de una información visual primaria a algo más elaborado. Con las características analizadas de cada región se debe de clasificar e interpretar. Por tanto, se diseñarán clasificadores que le dé a cada área segmentada una etiqueta de alto nivel, como por ejemplo, en una imagen aérea qué zonas son tierras de cultivo, áreas urbanas, etc. Existe un elenco de técnicas de clasificación, como redes neuronales, sistemas expertos, lógica borrosa, clasificadores estadísticos, etc. Éstas se verán muy someramente en el capítulo séptimo.

Otras presentaciones sobre las distintas etapas de la Visión Artificial son expuestas por otros autores. La más clásica es la dada por González y Woods mencionando tres tipos de nivel de información: bajo, medio y alto. La información de bajo nivel está dada por las etapas de adquisición y procesado, las de medio nivel son las de segmentación y extracción de las características y las de alto nivel con las etapas de reconocimiento e interpretación.

El valor añadido de esta presentación es la ubicación del conocimiento en el centro de todas estas etapas. Los desafíos del análisis de imágenes son extraordinariamente complejos y exigen de un conocimiento a priori sobre su problemática. La mayoría de las escenas que aborda la Visión Artificial son estructuradas, esto es, todos los elementos de iluminación están determinados y los objetos a capturar son previsibles. Por el contrario, una escena es no estructurada, cuando los objetos a visualizar son imprevisibles y la iluminación puede variar con el tiempo. Desde luego, la complejidad de los escenarios no estructurados se sale actualmente de la disciplina de la Visión Artificial. [3]

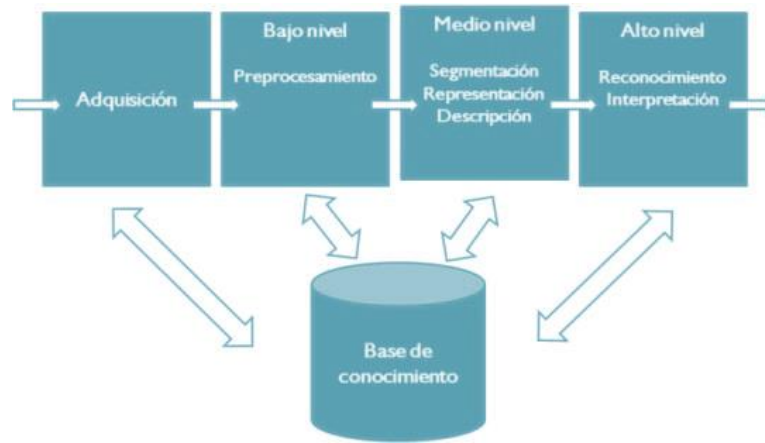


Figura 3. Etapas de la V.A. por niveles

1.4. Áreas de aplicación

Fotografía y Óptica: crear el ambiente de iluminación adecuada en la adquisición de las imágenes, muchas veces requiere del uso de técnicas profesionales de fotografía y vídeo. La selección de la óptica y de la cámara, el uso de filtros y polarizadores, las técnicas de iluminación con pantallas y la elección de los tipos de focos son algunas habilidades que se pueden mencionar.

Procesamiento Digital de las Imágenes: hace referencia a los algoritmos de computación que convierte la imagen digital adquirida en otra de mayor relevancia. Es muy difuso la separación entre el procesamiento de imágenes y la Visión Artificial.

Reconocimiento de Patrones: disciplina, dentro de la Inteligencia Artificial, dedicada a la clasificación de las señales y a la búsqueda de patrones existentes dentro de éstas. Se encuentran incluidas las técnicas de clasificadores estadísticos, Redes Neuronales, Sistemas Expertos, Lógica Borrosa.

Computación Gráfica: presenta el problema inverso de la Visión Artificial. Si en Visión se desea extraer las características físicas de las imágenes, la Computación Gráfica se dedica a la presentación visual de los modelos geométricos. Cada vez más, la Visión Artificial emplea la Computación Gráfica para representar las conclusiones extraídas del análisis de las imágenes adquiridas. [3]

1.5. Ejemplos de Visión Artificial

TABLA 2. EJEMPLOS DE VISIÓN ARTIFICIAL

Ejemplos	Descripción
Navegación en robótica	En este caso, la visión es un elemento de un sistema multisensorial. La información procedente de la visión es validada, comparada y finalmente integrada con el resto de la información proporcionada por otro tipo de sensores. [4]
La termografía como herramienta de diagnóstico en medicina deportiva	<p>La aplicación de la termografía en el campo de la medicina deportiva se encuentra actualmente en un interesante proceso de investigación. Gracias a la portabilidad de los equipos, la rapidez y escasez de riesgo de uso, estos sistemas pueden ser utilizados en clínicas especializadas o incluso en los mismos recintos de entrenamiento para una rápida evaluación del dolor y la consecuente toma de decisiones.</p> <p>La termografía puede resultar especialmente útil en la detección de dolores post-traumáticos crónicos, como la distrofia simpática refleja (DSR) o el síndrome de dolor simpático mantenido (DSM). [5]</p>
Image Pro Plus, un software eficaz para el estudio de la inervación del esófago	<p>En el Laboratorio de Malformaciones Congénitas del Servicio de Cirugía Pediátrica del Hospital Universitario de La Paz de Madrid utilizan Image-Pro Plus desde hace unos años.</p> <p>Recientemente han utilizado este software de visión para el estudio de la inervación del esófago de niños con hernia diafragmática congénita, una malformación congénita causada por un defecto en el cual el diafragma (el músculo grande en forma de cúpula que separa la cavidad torácica del abdomen) no logra desarrollarse completamente. [6]</p>
Scanner dental a través de imágenes 3D	A AQSENSE S.L. desarrolló una solución digitalizadora para la industria dental capaz de crear un modelo CAD a

	<p>partir de un molde dental, utilizando las herramientas SAL3D e a técnica de triangulos laser. El sistema proporciona una nueva línea de puntos organizada. El sistema completo permite unir e fazer coincidir diferentes vistas. É posible definir diferentes posiciones de barrido, de acuerdo con el área de interés y completar los procesos de Scanner en menos de 5 minutos. [7]</p>
Sistemas de seguridad y vigilancia	<p>Una imagen multiespectral consiste en imágenes de un mismo objeto, tomas con diferentes longitudes de onda. Puede ser luz visible, infrarrojos, ultravioleta, rayos-X u otra franja del espectro.</p> <p>Diversos aparatos hacen fotos espectrales. Pueden ser cámaras comunes de vigilancia o equipamientos de análisis para laboratorio, para análisis espectral. Pero comúnmente son asociados a satélites de detección remota y naves espaciales, pues muchos de ellos transportan cámaras multiespectrales. De esta forma pueden escoger que longitud registrar.</p> <p>Como ejemplo, la sonda Cassini consigue fotografiar los relieves de la nebulosa luna Tita, pues la luz visible es absorbida por su densa atmósfera. [8]</p>
Detección de caras	<p>Existen infinidad de aplicaciones de detección de rostros, tanto para seguridad como para otro tipo de aplicaciones, como por ejemplo en cámaras de fotos, para que el enfoque automático sepa donde tiene que enfocar.</p> <p>Como existe una gran cantidad de aplicaciones, el funcionamiento varía de unas a otras, pero independientemente de los patrones empleados por cada programa hay una serie de pasos comunes. [8]</p>
Control del tráfico	<p>Otra aplicación ampliamente utilizada es la detección de matrículas de los coches utilizando visión artificial. Existe un</p>

	<p>gran número de programas comerciales dedicados a esto y ampliamente utilizados en parkings, etc.</p> <p>En la imagen se puede apreciar una imagen del conocido programa “Visiomat”, muy extendido en parkings españoles. [8]</p>
Seguimiento de actividades humanas	<p>Otra aplicación que se está teniendo una gran expansión tanto para aplicaciones de vídeo-vigilancia como de interacción persona-computador es la del seguimiento de personas en secuencias de imágenes y la interpretación automática de las actividades que desarrollan.</p> <p>Para la realización de estas tareas se combinan métodos de seguimiento visual con el reconocimiento de patrones y aprendizaje. [8]</p>

2. Capítulo 2: Procesamiento digital de imágenes

2.1. Píxel.

Elemento más pequeño en que puede dividirse una imagen digital la superficie real que representa cada uno de ellos define los objetos o detalles más pequeños que pueden observarse en una imagen. El cual es utilizado por las cámaras por la limitación en la visión humana que lo percibe como un conjunto y no como unidades independientes Fig. 4.

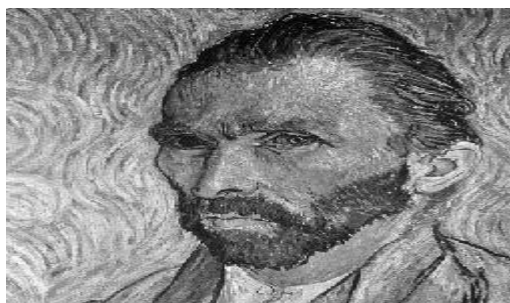


Figura 4. Imagen original

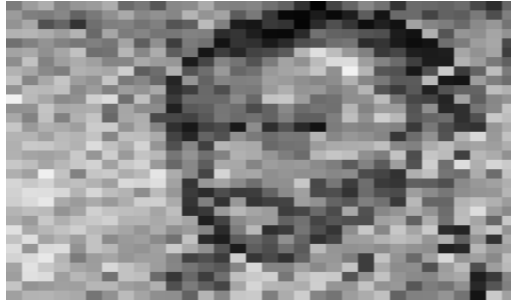


Figura 5. Zoom de la imagen [9]

2.2. Imagen digital

Una imagen digital consiste en una colección ordenada de valores. Estos valores se representan en una colección de filas de valores dispuestas ordenadamente.

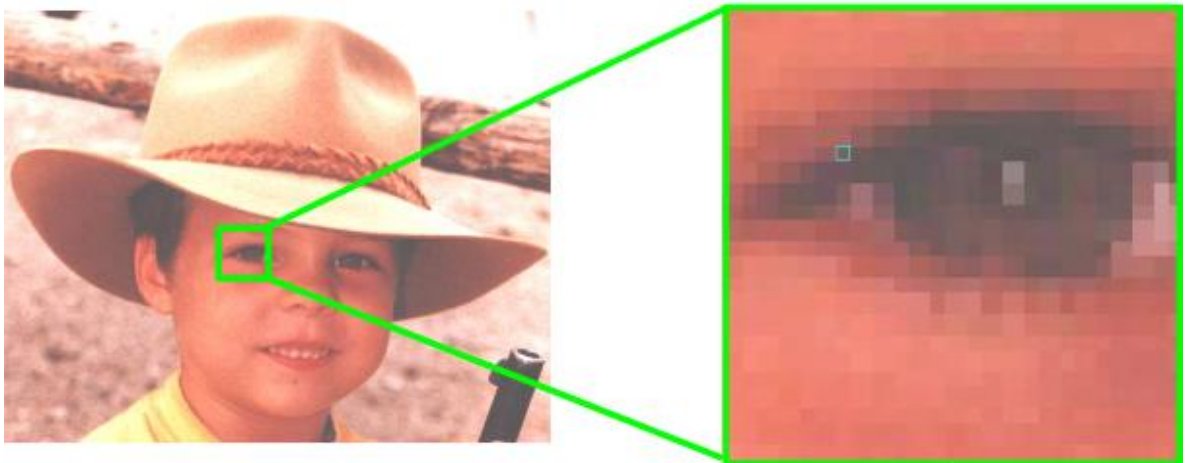


Figura 6. Representación imagen digital

¿Qué elementos definen una imagen digital?

- Tamaño de la imagen (medida en pixeles).
- Resolución de entrada (medida en pixeles o en dpi según el dispositivo).
- Resolución de salida (medida en dpi).
- Profundidad de color (medida en cantidad posible de colores).

- LUT (tabla de colores).
- Planos de color (RGB).
- Niveles de gris.
- Tamaño de fichero (medida en bytes).
- Tipo de fichero (formato en el que se ha guardado). [10]

Histograma

Un histograma es un gráfico que muestra la distribución de los colores o tonos de un color en una imagen según su luminosidad. En el histograma, el eje horizontal indica la luminosidad (más a la izquierda, más oscuro y más a la derecha, más luminoso). El eje vertical indica la cantidad de píxeles con esa luminosidad. Un pico en nuestro histograma en el lado izquierdo indica un gran número de píxeles que están oscuros o negros (posiblemente una foto subexpuesta) mientras que un pico en la parte derecha indica un gran número de luminosos o blancos (posiblemente una foto sobreexpuesta). Por este razonamiento un histograma uniforme (sin picos) en todos los tonos es probable que indique que la imagen está debidamente expuesta.

La Fig. 6 y 7 nos muestran una imagen monocromática y su histograma. [11]

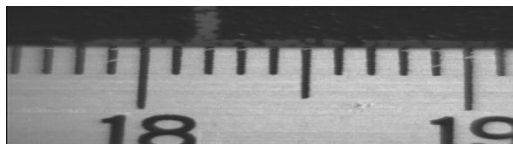


Figura 7. Imagen monocroma codificada en 256 niveles de intensidad (niveles de gris)

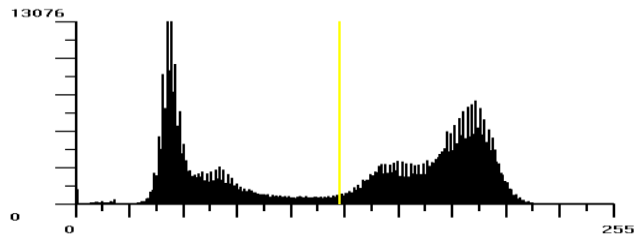


Figura 8. La línea amarilla en el histograma indica el valor medio de intensidad.

2.3. Clasificación de imágenes digitales

2.3.1. Imágenes binarias

Las imágenes digitales generalmente están compuestas por un amplio rango de valores de intensidad. A estas imágenes se las ha denominado imágenes de nivel de gris. Aunque el rango de niveles de gris utilizado para representar la imagen tradicionalmente viene siendo de 256 valores (8 bits por píxel), este es variable y depende de la aplicación. La tendencia natural, debido al aumento de la potencia computacional y de la calidad de los sensores es a aumentar este rango para dotar de mayor fidelidad a la imagen, y no parece que esté muy lejos el día en que los 16 bits por píxel (65536 valores) se conviertan en la opción estándar. No obstante, la mayor parte de las aplicaciones no precisan de tantos niveles de gris sino más bien lo contrario; pueden utilizar pocos niveles debido a que trabajan con escenas de muy alto contraste. Tanto es así que en muchas aplicaciones industriales se llega al extremo de utilizar únicamente dos niveles de gris. De esta forma, se obtiene lo que se conoce como imagen binaria.

Trabajar con imágenes binarias resulta muy interesante por dos motivos:

- En primer lugar porque se reduce al mínimo los datos necesarios para representar la imagen y ello permite un máximo aprovechamiento de la potencia computacional.
- En segundo lugar porque las propiedades geométricas y topológicas de los objetos presentes en la imagen, en las que se basan un gran número de aplicaciones industriales, puede obtenerse rápida y fácilmente a partir de las imágenes binarias.

Desde el punto de vista computacional, las imágenes binarias se procesan mucho más rápidamente. Por este motivo, las primeras aplicaciones de visión artificial en línea emplearon este tipo de imágenes casi exclusivamente. Incluso hoy en día, que se dispone de potentes procesadores para hacer frente a imágenes en niveles de gris, siguen siendo aún más numerosas las aplicaciones que trabajan sobre imágenes binarias por su simplicidad y robustez. Dado el protagonismo que tienen en el ámbito industrial las dedicaremos un capítulo en exclusiva.

Las imágenes binarias siempre se obtienen a partir de imágenes de niveles de gris. En la actualidad no existen cámaras comerciales que proporcionen imágenes binarias. El proceso de conversión de una imagen de nivel de gris a una imagen formada solo por dos valores o etiquetas (0 para el negro y 1 para el blanco) se conoce como binarización [12].

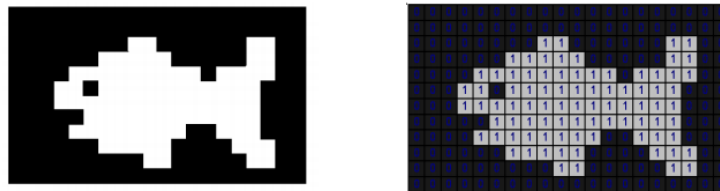


Figura 9. Binarización de una imagen

2.1.1. Imágenes de intensidad.

Para este modo cada pixel puede adoptar distintas gamas de grises, las cuales pueden tomar valores enteros comprendidos entre 0 y 255. En este caso, al color negro se le asigna el valor “0” mientras que al blanco el valor “255”. Por ejemplo, un pixel de color gris al 20 % (es decir, 20% negro y 80% blanco) le corresponde el valor 204, mientras que a uno de color gris al 50 % le corresponde el valor 128. [13]

En Escala de Grises se define una imagen digital de la siguiente manera:

$$f: A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1, 2, \dots, 255\}$$

Dada la imagen digital en Escala de Grises de 3 x 3 píxeles definida por

$$f(1,1) = 128 \quad f(2,1) = 0 \quad f(3,1) = 204$$

$$f(1,2) = 0 \quad f(2,2) = 255 \quad f(3,2) = 0$$

$$f(1,3) = 204 \quad f(2,3) = 0 \quad f(3,3) = 128$$

Su representación gráfica es la siguiente:

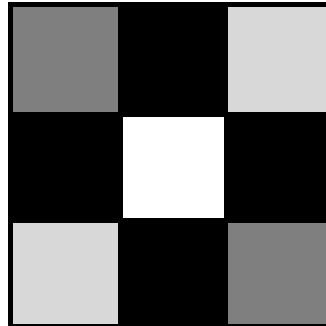


Figura 10. Imagen en escala de grises

2.1.2. Imágenes en color RGB.

Es el modo estándar para la representación de imágenes en pantallas. Su nombre proviene de la unión de las primeras letras de las palabras inglesas **R**ed, **G**reen y **B**lue.

En este modo, cada pixel puede tomar un color que resulta de la combinación de las distintas tonalidades de rojo, verde y azul. De esta manera, a cada pixel se le asocia un vector de 3 componentes **(R, G, B)**, en las que cada una de ellas puede tomar valores enteros comprendidos entre 0 y 255. Las componentes **R**, **G** y **B** representan la cantidad de rojo, verde y azul respectivamente.

En modo RGB una imagen digital queda definida del siguiente modo:

$$f: A \subset \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1, \dots, 255\} \times \{0, 1, \dots, 255\} \times \{0, 1, \dots, 255\}$$

Por ejemplo, a un pixel de color rojo al 100% le corresponde el vector (255, 0, 0), a uno de color verde al 100% le corresponde el vector (0, 255, 0), mientras que a uno de color turquesa el vector (0, 255, 255). En RGB, todas las componentes en 0 forman el negro,

mientras que todas las componentes en 255 forman el blanco. Combinando los distintos valores de rojo, verde y azul, un pixel puede adoptar $256^3 = 16.777.216$ colores diferentes.

2.4. Formatos de imágenes.

De manera general existen dos tipos de imágenes digitales; Imágenes vectoriales y las imágenes de mapas de bits.

Las **imágenes vectoriales** son imágenes constituidas por objetos geométricos autónomos (líneas, curvas, polígonos), definidos por ciertas funciones matemáticas (vectores) que determinan sus características (forma, color, posición).

Las **imágenes de mapa de bits** están formadas por una serie de puntos (píxeles), cada uno de los cuales contiene información de color y luminosidad.



Figura 11. Dos formatos de imágenes digitales.

Las imágenes vectoriales se crean con programas de diseño o dibujo vectorial (Adobe Ilustrador, Corel Draw, Inkscape...) y suelen usarse en dibujos, rótulos, logotipos. Su principal ventaja es que una imagen puede ampliarse sin sufrir el efecto de “pixelado” que tienen las imágenes de mapa de bits al aumentarse.



Figura 12. Ampliación de imágenes.

En la imagen (vectorial) del ratón de la izquierda puede apreciarse que al ampliar una zona no hay pérdida de detalle, mientras que en la fotografía del busto Nefertiti (mapa de bits) al ampliar mucho una zona, se observan los píxeles y la imagen se degrada.

Las imágenes de mapa de bits presentan una mayor gama de colores y de tonos que las vectoriales, por lo que son el tipo de imágenes usado en fotografía, se crean con las (Microsoft).

Algunos formatos de mapa de bits son los siguientes: cámaras de fotos, los escáneres y con programas de edición de imagen y dibujo (Adobe Photoshop, Gimp, etc.) Las imágenes mapa de bits generan archivos que ocupen mucha más memoria (bytes) que las imágenes vectoriales.

Para poder reproducirse o utilizarse en un ordenador u otros dispositivos las imágenes vectoriales y de mapa de bits se guardan en archivos o ficheros (conjunto de datos que se almacenan en algún medio, como un disco duro, DVD, flash memory) Cada archivo gráfico, se identifica además de por su nombre, por su extensión, que indica el tipo o formato de que se trata.

Algunos formatos de imagen vectorial son: AI (Adobe Illustrator), CDR (Corel Draw), DXF (Autodesk), EPS, ODG (Open Office Draw), SVG (Inkscape), SWF (Adobe flash), WMF.

- **BMP.** Formato introducido por Microsoft y usado originariamente por el sistema operativo Windows para guardar sus imágenes.
- **GIF.** Formato bastante antiguo desarrollado por Compuserve con el fin de conseguir archivos de tamaño muy pequeños. Admite solo 256 colores por lo que no es adecuado para imágenes fotográficas pero si es muy apropiado para logotipos, dibujos, etc. Permite crear animaciones (gif animado) y transparencias.
- **JPEG.** Es uno de los formatos más conocido y usado para fotografías digitales ya que admite millones de colores. Lo admiten la mayor parte de las cámaras fotográficas y escáneres y es muy utilizado en páginas web, envío de fotografías por correo electrónico, presentaciones multimedia y elaboración de vídeos de fotografías.
- **PNG.** Formato creado con el fin de sustituir a GIF. Utiliza sistemas de compresión gratuitos, y admite muchos más colores que GIF. También admite transparencias pero no animaciones. Al admitir más colores es posible crear imágenes transparentes con mayor detalle.
- **RAW.** Formato “en bruto”. Esto quiere decir que contiene todos los píxeles de la imagen captada, tal y como se han tomado. Es el formato que ofrece la mayor calidad fotográfica y suele ser admitido por cámaras de gama media y alta (réflex, y compactas) indicadas para fotógrafos aficionados avanzados y profesionales.

Las cámaras que guardan las fotos en otros formatos (Tiff y JPEG) procesan la imagen captada para dar una interpretación de ella (balance de blanco, niveles de luminosidad, contraste) En el formato RAW, los píxeles no se procesan y se mantienen en bruto para ser procesados posteriormente por un software específico conocido como “revelador RAW”.

TIFF. Formato utilizado para el escaneado, la edición e impresión de imágenes fotográficas Es compatible con casi todos los sistemas operativos y editores de imágenes. Como PSD, admite millones de colores, capas, canales alfa... y también lo incluyen algunas cámaras y la mayoría de los escáneres [14].

2.5. Filtros más utilizados en el procesamiento de imágenes.

2.5.1. Bordes de Sobel.

El filtro Sobel detecta los bordes horizontales y verticales separadamente sobre una imagen en escala de grises. Las imágenes en color se convierten en RGB en niveles de grises. Como con el filtro Laplace, el resultado es una imagen transparente con líneas negras y algunos restos de color [15].

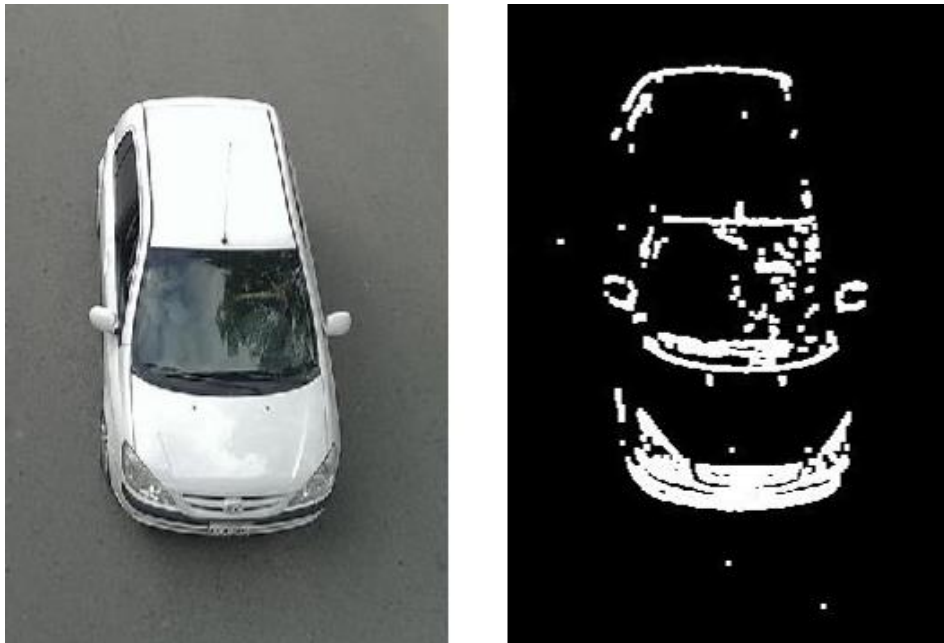


Figura 13. Imagen original e imagen con filtro de sobel

2.5.2. Bordes de Canny.

El algoritmo de Canny consiste en tres grandes pasos:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.
- Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.

- Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos [16].

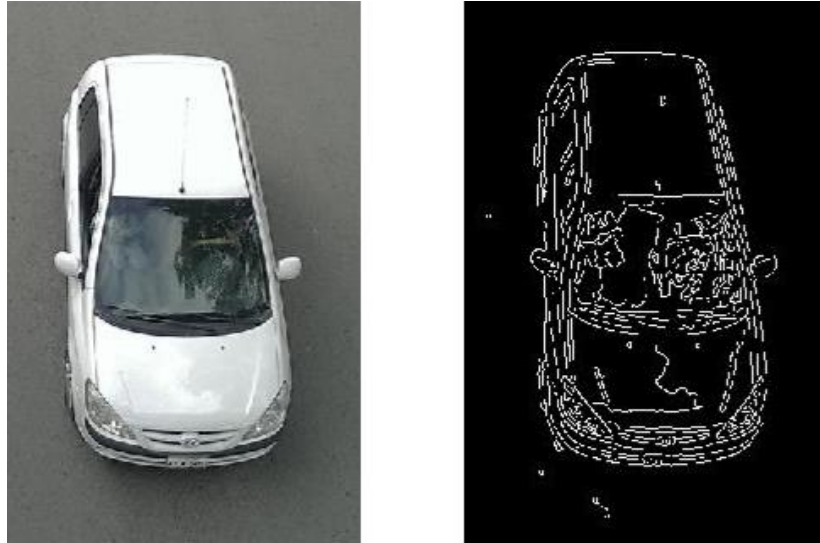


Figura 14. Imagen original e imagen con filtro de Canny

2.5.3. Bordes de Prewitt.

Para cada pixel de la imagen se calcula un valor a partir de los pixels vecinos. Se suele denominar que se aplica (convoluciona) una máscara, así en el caso de Prewitt se utilizan dos máscaras una en el sentido de las X y otra en el sentido de las Y (se considera que la imagen es una matriz) y después se calcula el módulo de los gradientes direccionales [17].

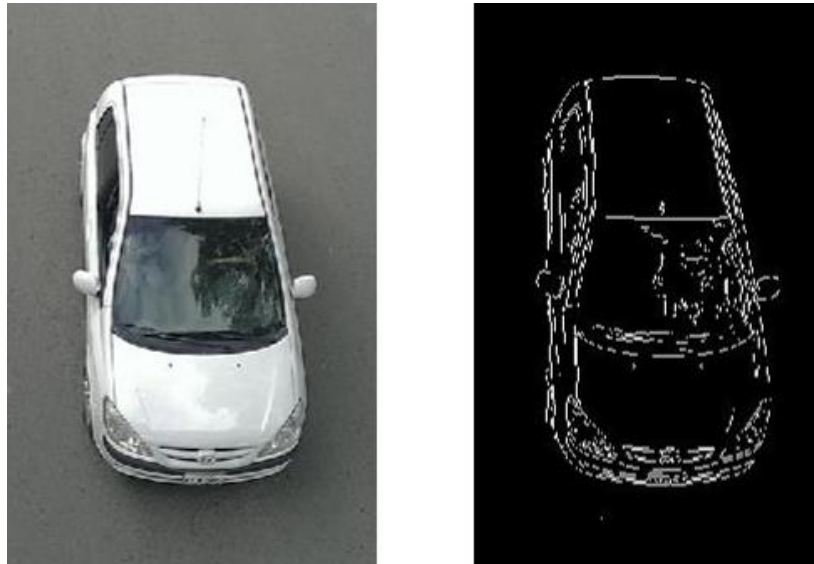


Figura 15. Imagen original e imagen con filtro de prewitt.

2.5.4. Bordes de Roberts.

Obtiene buena respuesta ante bordes diagonales. Ofrece buenas prestaciones en cuanto a localización. El gran inconveniente de este operador es su extremada sensibilidad al ruido y por tanto tiene pobres cualidades de detección [18].

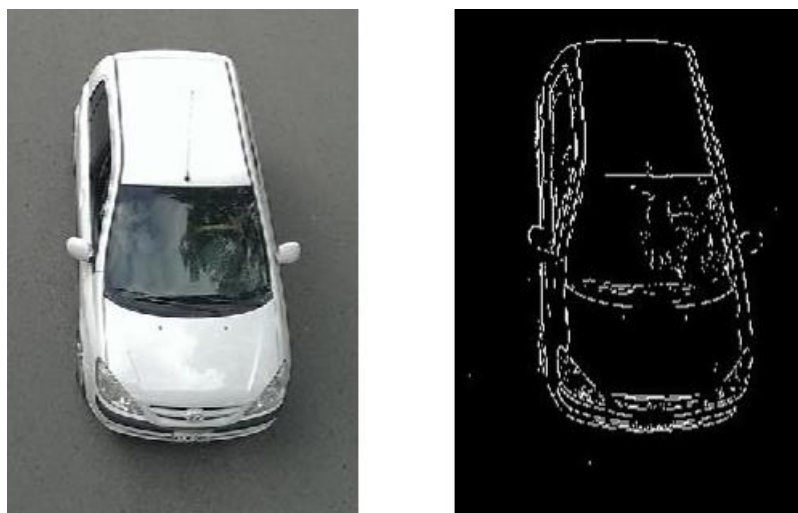


Figura 16. Imagen original e imagen con filtro de Roberts.

2.6. Técnicas para el procesamiento de imágenes.

2.6.1. Escala de grises.

Luminosidad o intensidad del píxel que va de 0 (negro) a 255 (blanco). El tono de gris de cada píxel se puede obtener bien asignándole un valor de brillo que va de 0 (negro) a 255 (blanco) bien como porcentajes de tinta negra (0% es igual a blanco y 100% es igual a negro). Las imágenes producidas con escáneres en blanco y negro o en escala de grises se visualizan normalmente en el modo escala de grises. Este modo maneja un solo canal (el negro) para trabajar con imágenes monocromáticas de 256 tonos de gris entre el blanco y el negro [19].



Figura 17. Imagen en modo escala de grises

2.6.2. Binarización.

Es una variante de la Umbralización crea una imagen de salida binaria a partir de una imagen de grises donde todos los valores de gris cuyo nivel está en el intervalo definido por p_1 y p_2 son transformados a 255 y todos los valores fuera de ese intervalo a 0 [19].

$$q = \begin{cases} 255 & \text{para } p \leq p_1 \text{ ó } p \geq p_2 \\ 0 & \text{para } p_1 < p < p_2 \end{cases}$$

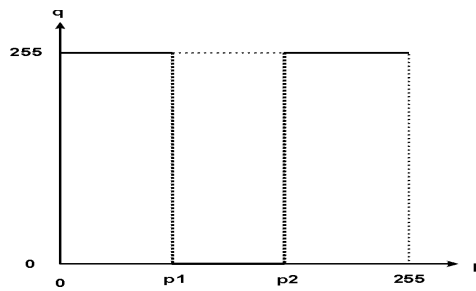


Figura 18. Función de transformación y operador intervalo de umbral binario



Figura 19. Imagen original e imagen binarizada

2.6.3. Umbralización.

También conocida como *thresholding* es una técnica de segmentación de imágenes en la que cada píxel pertenece obligatoriamente a un segmento y sólo uno. Si el nivel de gris del píxel es menor o igual al umbral se pone a cero si es mayor se pone a 255. En función del valor umbral que se escoja el tamaño de los objetos irá oscilando [19].

$$q = \begin{cases} 0 & \text{para } p \leq p_1 \\ 255 & \text{para } p > p_1 \end{cases}$$

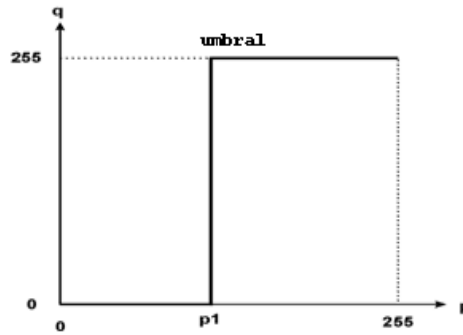


Figura 20. Función de transformación y grafica del operador umbral.

La Fig. 20 muestra la oscilación de la calidad de la imagen con respecto a un valor de umbral.



Figura 21. Imagen original y segmentación con umbral de 128

2.6.4. Redes neuronales

Las redes de neuronas artificiales llamadas Perceptrones Multicapas (PM) son una herramienta atractiva para solucionar problemas de clasificación como el reconocimiento

de caracteres manuscritos, el reconocimiento de palabras habladas, y el diagnóstico de diferentes enfermedades [20].

Las Redes Neuronales son un campo muy importante dentro de la Inteligencia Artificial. Inspirándose en el comportamiento conocido del cerebro humano (principalmente el referido a las neuronas y sus conexiones), trata de crear modelos artificiales que solucionen problemas difíciles de resolver mediante técnicas algorítmicas convencionales.

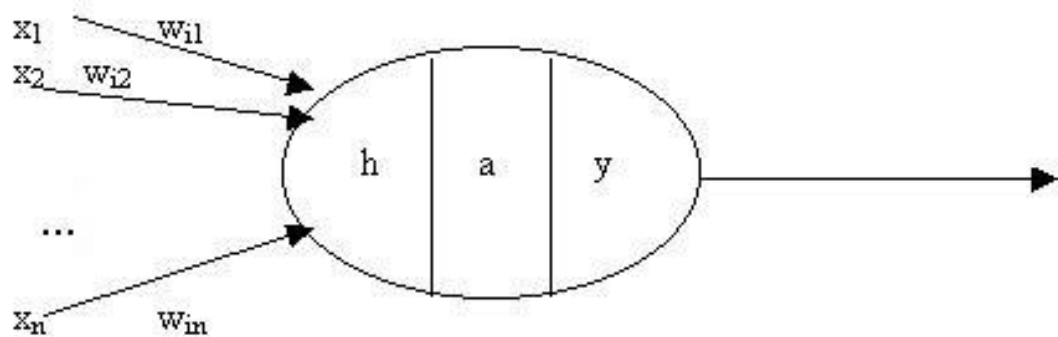


Figura 22. Modelo de una neurona artificial

Esta neurona artificial consta de los siguientes elementos:

- Conjunto de entradas o vector de entradas x , de n componentes
- Conjunto de pesos sinápticos w_{ij} . Representan la interacción entre la neurona presináptica j y la postsináptica i .
- Regla de propagación $d(w_{ij}, x_j(t))$: proporciona el potencial postsináptico, $h_i(t)$.
- Función de activación $a_i(t)=f(a_i(t-1), h_i(t))$: proporciona el estado de activación de la neurona en función del estado anterior y del valor postsináptico.
- Función de salida $F_i(t)$: proporciona la salida $y_i(t)$, en función del estado de activación [21].

2.6.5. Viola Jones

El algoritmo de Viola-Jones permite la detección robusta en tiempo real de caras. El algoritmo de Viola-Jones supone un gran avance dentro de los algoritmos de detección por su gran rapidez, y porque la clasificación se realiza mediante características en vez de píxel a píxel, lo que permite una cierta abstracción del algoritmo respecto el resultado.

Este estudio permitió desarrollar potentes clasificadores en otras áreas de la vision artificial, tal cual es el caso de la medicina, industria y transporte [22].

2.6.5.1. Haar-like features

La principal razón para usar características en el algoritmo es que permite hacer una asociación entre conocimiento aprendido y el análisis de los datos adquiridos. Además, la clasificación por características es mucho más rápida que el procesado por análisis basados en píxel.

Las características usadas son las mismas que fueron usadas por Papageorgiou et al. (1998)³. Las características de Haar (*Haar-like features* en inglés) permiten obtener información de una zona concreta mediante una operación aritmética simple: Esto nos lleva a una gran eficiencia tanto de cálculo como espacial [22].

En concreto se usan tres características de Haar:

- Característica de dos rectángulos: es la diferencia entra la suma de los píxeles de ambas regiones rectangulares. Las regiones tienen el mismo tamaño y forma y están horizontalmente o verticalmente adyacentes.
- Característica de tres rectángulos: es la suma de los píxeles en ambos rectángulos exteriores sustraídos por la suma en el rectángulo central.
- Característica de cuatro rectángulos: es la diferencia entre los pares diagonales de los rectángulos.

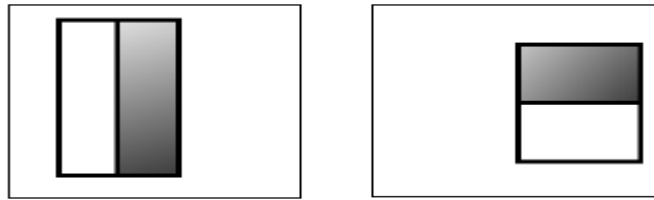


Figura 23. Características de 2 rectángulos

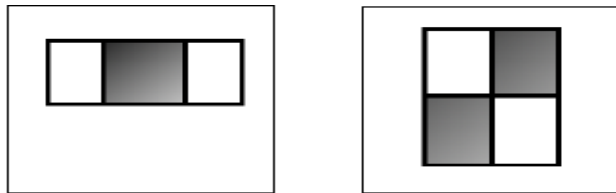


Figura 24. Características de 3 y 4 rectángulos

En todos los casos anteriores la obtención del valor de la característica consiste en la resta entre el valor de una o más subzonas dentro de la zona analizada. Es decir, restamos el valor que damos una subzona, mediante la integral sumada (explicada más adelante en este mismo documento), con el de otra subzona.

De manera simplificada, las características pueden ser vistas como evaluaciones de la intensidad de conjuntos de píxeles. La suma de la luminancia de los píxeles en la región blanca se resta de la suma de los píxeles en la región oscura. El valor obtenido mediante la diferencia es el valor de la característica las cuales pueden estar en cualquier posición y escala de la imagen original y puede combinarse con otros formando hipótesis en regiones de una imagen. Cada tipo de característica puede indicar la existencia o no de un tipo de características en la imagen [19].

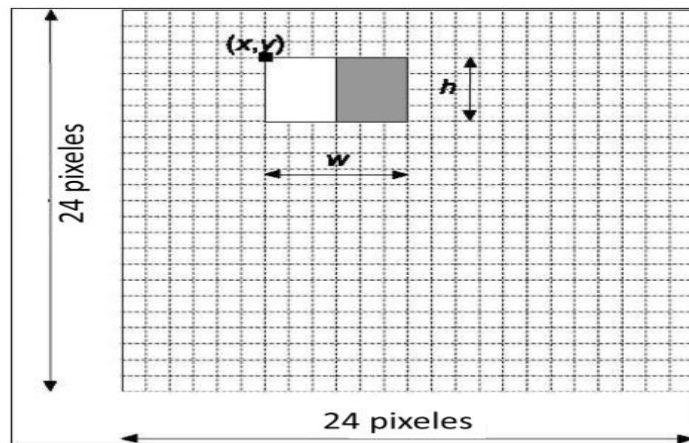


Figura 25. Representación de una imagen genérica con un modelo de característica.

2.6.5.2. Imagen Integral

A la hora de crear un sistema de detección resulta crucial encontrar un compromiso entre velocidad y eficiencia. Mediante el uso de una nueva representación de las imágenes, llamada imagen integral, Viola y Jones describen un método de evaluación de características de manera efectiva y a mayor velocidad.

La imagen integral es una matriz del mismo tamaño que la matriz de la imagen original donde cada elemento de la Imagen Integral a la posición $(x; y)$ contiene la suma de todos los píxeles localizados en la región superior izquierda de la imagen original.

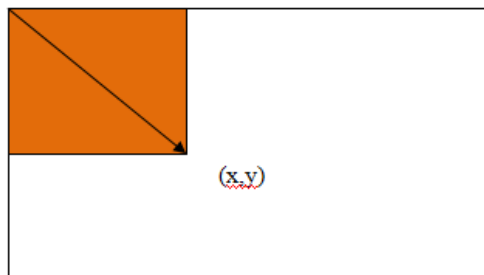


Figura 26. Valor de la imagen integral en un punto (x, y) .

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

$ii(x, y)$ es la imagen integral y $i(x, y)$ es la imagen original. Usando las dos siguientes operaciones:

$$S(x, y) = S(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + S(x, y)$$

Donde $S(x, y)$ es suma acumulada en línea y $S(x, -1) = 0$ y $ii(-1, y) = 0$

Así la imagen integral puede ser calculada en un solo barrido sobre la imagen original.

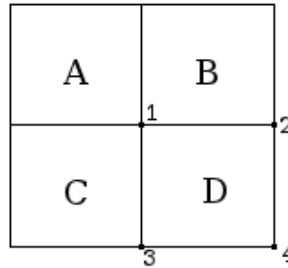


Figura 27. Resultado de la imagen integral

Cualquier suma dentro de un área de la imagen puede calcularse utilizando cuatro referencias. Por lo tanto la diferencia entre dos regiones puede calcularse utilizando 8 referencias dentro de la imagen. Sin embargo, teniendo en cuenta que, por ejemplo los dos primeros tipos de características utilizan dos regiones rectangulares adyacentes la diferencia puede realizarse utilizando 6 referencias, en el caso del tercer tipo se utilizarían 8 referencias y en el cuarto tipo 9 referencia.

2.6.5.3. El clasificador

Una vez encontradas las características dentro de una imagen, el objetivo del sistema es encontrar aquellas características que mejor definan una cara o parte superior del cuerpo y

nos ayuden a localizarla en frame de video. La hipótesis planteada en las publicaciones de Viola y Jones establece que un pequeño número de esas características pueden combinarse para formar un clasificador. El algoritmo se utiliza para mejorar el rendimiento de un algoritmo de aprendizaje simple. Este algoritmo simple se llama clasificador simple o *weak learner*. [23]

2.6.5.4. Clasificador simple

La base del sistema de Viola y Jones es la combinación de clasificadores simples o *weak learners* para obtener una clasificación eficiente de los datos. En relación a esto, un *weak learner*, h_j , es una estructura simple que contiene una característica f , un umbral θ y una paridad p . La salida del clasificador es binaria y depende de si el valor de una característica se encuentra por encima o debajo de un umbral.

$$h(x, f, p, \theta) \begin{cases} 1 & \text{si, } pf(x) < p\theta \\ 0 & \text{en caso contrario} \end{cases}$$

Los clasificadores simples que se utilizan pueden verse como nodos de decisión en estructuras en árbol. Teniendo en cuenta que hay una gran cantidad de características en una imagen, el algoritmo AdaBoost debe seleccionar las características que mejor diferencien entre el objeto a detectar (caras y no caras o parte superior del cuerpo y no parte superior del cuerpo dentro de un frame de video, y en él recalca la mayor parte del trabajo del entrenador. El problema que se plantea a la hora de realizar una clasificación consiste en decidir a qué “*clase*” pertenece un objeto. [23].

2.6.5.5. Algoritmo Adaboost

El algoritmo de aprendizaje Adaboost, que significa “adaptive boosting”, se empleó para elegir los mejores clasificadores simples y formar la función de clasificación. Es uno de los algoritmos más utilizados en aprendizaje automático. La ventaja principal de AdaBoost es su velocidad de aprendizaje [22].

En el algoritmo de Viola-Jones el algoritmo AdaBoost es capaz de elegir entre un gran conjunto de filtros, las características de Haar, para seleccionar en cada momento cuál de

ellos se ajusta mejor para que se clasifique satisfactoriamente los diferentes elementos que queremos clasificar.

En la Tabla 3. se muestra el algoritmo de Adaboost desarrollado y publicado por sus autores Viola y Jones. [24]

TABLA 3. ALGORITMO DE ADABOOST

- Dado un conjunto de imágenes $(x_1, y_1), \dots, (x_n, y_n)$ donde $y_1 = 0, 1$ para muestras negativas y positivas respectivamente.
• Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i=0,1$
Donde m es el número de muestras negativas y l es el número de muestras positivas.
• Para $t = 1, \dots, T$:
1) Normalizar los pesos $W_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$
2) Seleccionar la mejor clasificadora base respecto al error de peso.
$\epsilon_t = \min_{f,p,\theta} \sum_i w_i h(x_i, f, p, \theta) - y_i $
3) Define $h_t(x) = h(x, f_t, p_t, \theta_t)$, donde f_t, p_t, θ_t son usadas para minimizar ϵ_t
4) Actualiza los pesos: $W_{t+1,i} = W_{t,i} \beta_t^{1-e_i}$
Donde $e_i = 0$ si la muestra x_i es clasificada correctamente, o $e_i = 1$ en otro caso,
con $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
• El clasificador robusto final queda: $C(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$
Donde $\alpha_t = \log \frac{1}{\beta_t}$

Con este procedimiento obtenemos que después de algunas iteraciones, donde hemos probados todos los clasificadores de Haar para cada una de las muestras, tenemos un clasificador que separa acordemente entre muestras positivas y muestras negativas.

En un primer paso, se genera una detección con algún clasificador. Una vez hecho esto los elementos mal clasificados aumentan su peso para que el siguiente clasificador usado de más importancia a que la clasificación de los elementos con mayor peso sea la correcta.

Una vez realizado este último paso con diferentes clasificadores, obtenemos un único clasificador como combinación de los anteriores y que clasifica todos los elementos correctamente con un valor de error aceptable.

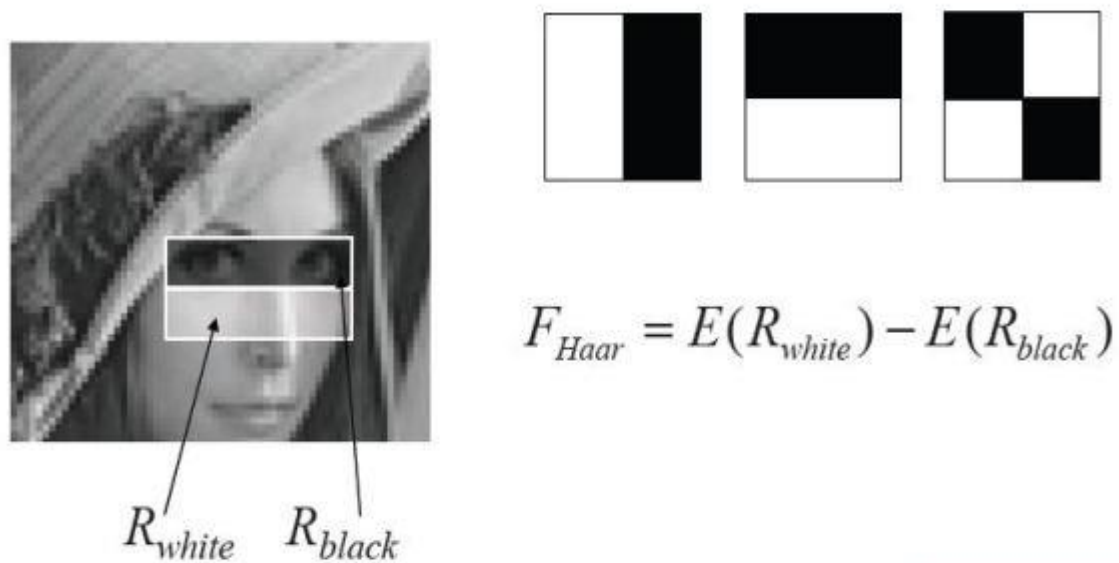


Figura 28. Descriptores rectángulos

2.6.5.6. El detector de clasificadores en cascada

El principio básico del algoritmo de detección facial de Viola y Jones consiste en escanear el detector muchas veces a través de una misma frame, en diferentes posiciones y a distintas escalas. Incluso si una imagen contiene muchas caras está claro que la mayoría de las sub-ventanas que se escaneen no contendrán ninguna cara. Esto lleva a una nueva manera de ver el problema: *En vez de encontrar caras, el algoritmo debería descartar „no-caras“*.

La idea subyacente se basa en que es más fácil descartar un frame que no contenga una cara que encontrar una cara en un frame. Con esto en la cabeza, parece ineficiente construir un detector que contenga un solo clasificador fuerte ya que el tiempo de clasificación será constante sin importarnos la entrada del clasificador, ya que el clasificador tiene que evaluar todas las características que lo forman. Aumentar la velocidad de clasificación generalmente implica que el error de clasificación aumentará inevitablemente, ya que para disminuir el tiempo de clasificación se debería disminuir el número de clasificadores simples que se utilizan. Para evitar esto Viola y Jones proponen un método para reducir el tiempo de clasificación manteniendo los requerimientos de rendimiento del clasificador [57].

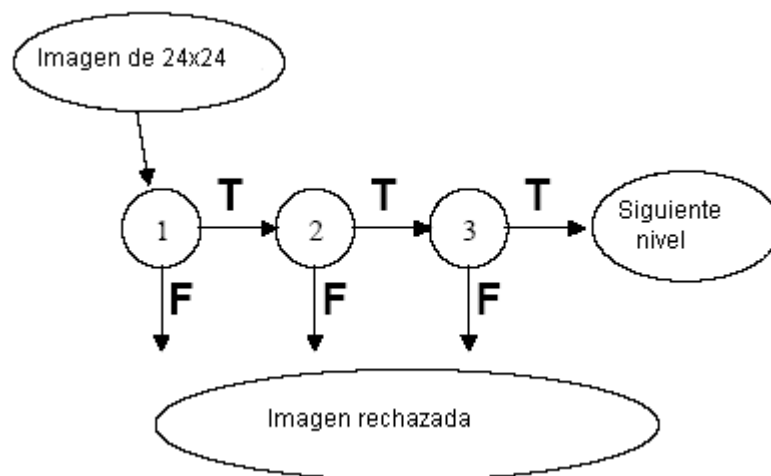


Figura 29. Estructura de clasificadores en cascada

Este método consiste en el uso de una cascada de clasificadores fuertes. El trabajo en cada etapa del clasificador consiste en determinar si la sub-ventana que se analiza es definitivamente un valor positivo. Cuando una sub-ventana es clasificada como un valor negativo en alguna de las etapas del detector, se descarta inmediatamente. En caso contrario, si se clasifica como un posible positivo, pasa a la siguiente etapa del clasificador. (Ver Figura 29.).

Si se utilizase un clasificador que tuviese un único estado, normalmente habría que aceptar los falsos negativos para reducir la tasa de falsos positivos. Sin embargo, para las primeras etapas del clasificador en cascada se acepta una alta tasa de falsos positivos esperando que las etapas posteriores puedan encargarse de reducir esta tasa mediante clasificadores más especializados. Con esto se pretende también reducir la tasa de falsos negativos en el clasificador final.

2.6.5.7. Entrenamiento de la cascada

El proceso de diseño de la cascada empleado por Viola y Jones se basa en objetivos de detección y rendimiento similares a lo que se hizo hasta el momento. Los sistemas anteriores obtenían tasas de detección de entre el 85% y el 95% y tasas de falsos positivos extremadamente bajas del orden de 10^{-5} . Es por ello que el sistema debería tener un número de etapas suficientes para obtener resultados similares.

Dada una cascada de clasificadores, la tasa de falsos positivos se calcularía como sigue:

$$F = \prod_{i=1}^K f_i$$

Donde F es la tasa de falsos positivos del detector, K es el número de clasificadores, y f_i es la tasa de falsos positivos calculada para el clasificador i. Por otro lado la tasa de detección de la cascada se calcularía según:

$$D = \prod_{i=1}^K d_i$$

Donde D es la tasa de detección del detector, K es el número de clasificadores, y d_i es la tasa de detección calculada para el clasificador- i . Una vez establecido esto, vemos como por ejemplo un detector consistente en 10 etapas con una tasa de detección del 99% por etapa y una tasa de falsos positivos del 30% por clasificador nos llevaría a una tasa de detección global del $0.99^{10} \approx 0.9$ y una tasa de falsos positivos global del $0.3^{10} \approx 6 \times 10^{-6}$.

El proceso de entrenamiento debe considerar las limitaciones a las que se ve sometido. En la mayoría de los casos, los clasificadores construidos utilizando más características tendrán una mayor tasa de detección y una menor tasa de falsos positivos. Al mismo tiempo, los clasificadores con más características necesitarán más tiempo para determinar si una sub-ventana contiene o no una cara. A la hora de entrenar el clasificador uno debe optimizar el número de etapas del clasificador, el número de características y el umbral de cada etapa. Sin embargo, realizar esta optimización es un trabajo tremendamente costoso.

Para simplificar este problema los autores han realizado un algoritmo para poder entrenar de manera efectiva la cascada de clasificadores. En este caso el usuario elige las tasas de falsos positivos y de detección de cada etapa así como la tasa de falsos positivos referente a todo el detector. Cada etapa del detector se entrena utilizando AdaBoost del modo descrito en la Tabla V, incrementando el número de características de la etapa hasta que se obtengan las tasas de falsos positivos y de detección deseadas. Dichas tasas se determinan confrontando el detector con un set de validación. Si la tasa de falsos positivos global no es la que interesa se añade otra etapa al clasificador. [24]

3. Capítulo 3. Herramientas, técnicas y librerías utilizadas en el desarrollo del sistema.

3.1. Herramientas

Al momento de desarrollar aplicaciones que estén basadas en Visión Artificial es muy importante saber elegir una librería como herramienta de desarrollo, aunque no sería muy recomendable utilizar solo una. Es probable que las deficiencias de una puedan ser resueltas por otra, o que para un problema concreto resulte aconsejable usar una librería específica como es el caso de Opencv.

A continuación se detallan algunas de las librerías más utilizadas en el desarrollo de aplicaciones que estén enfocadas en la Visión Artificial.

3.1.1 Cámara

3.1.1.1. Cámara digital

Una cámara digital es una cámara fotográfica que, en vez de captar y almacenar fotografías en película química como las cámaras fotográficas de película fotográfica, recurre a la fotografía digital para generar y almacenar imágenes. [25]

3.1.1.2. Cámara ip

Una cámara IP es una cámara que emite las imágenes directamente a la red (intranet o internet) sin necesidad de un ordenador. [26]

3.1.1.3. Cámara usb

Una cámara usb es aquella que para transmitir los datos necesita de la conexión por puerto usb de un computador.

3.1.1.4. Cámara integrada en celular

Es aquella cámara que se integra en el hardware de un teléfono móvil, hoy en día muchas de estas cámaras son en el mejor de los casos mejores que algunas cámaras digitales.

3.1.2. Ubuntu



Figura 30. Ubuntu. Última versión disponible 14.04 LTS

Ubuntu es una palabra Africana que significa 'Humanidad hacia otros', o 'Yo soy porque nosotros somos'. La distribución Ubuntu lleva el espíritu de Ubuntu al mundo del software.

Ubuntu es un sistema operativo desarrollado por la comunidad que es perfecto para laptops, computadoras de escritorio y servidores. Ya sea que lo utilices en el hogar, en la escuela o en el trabajo, Ubuntu contiene todas las aplicaciones que puedas necesitar, desde procesadores de texto y aplicaciones de email, hasta software para servidor web y herramientas de programación.

Ubuntu es y siempre será libre de costo. No pagas por una licencia de uso. Puedes descargar, usar y compartir Ubuntu con tus amigos, familiares, escuela o negocios libremente [27].

Su patrocinador, Canonical, es una compañía británica propiedad del empresario sudafricano Mark Shuttleworth. Ofrece el sistema de manera gratuita, y se financia por medio de servicios vinculados al sistema operativo y vendiendo soporte técnico. Además, al mantenerlo libre y gratuito, la empresa es capaz de aprovechar los desarrolladores de la comunidad para mejorar los componentes de su sistema operativo. Extraoficialmente, la comunidad de desarrolladores proporciona soporte para otras derivaciones de Ubuntu, con otros entornos gráficos, como Kubuntu, Xubuntu, Ubuntu MATE, Edubuntu, Ubuntu Studio, Mythbuntu, Ubuntu GNOME y Lubuntu [28].

3.1.3. Lenguaje C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

3.1.4. Qt



Figura 31. Qt, Última versión 4.8

Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores. Es desarrollada como un software libre y de código abierto a través de Qt Project, donde participa tanto la comunidad, como desarrolladores de Nokia, Digia y otras empresas. Anteriormente, era desarrollado por la división de software de Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008. Qt es distribuida bajo los términos de GNU Lesser General Public License (y otras). Por otro lado, Digia está a cargo de las licencias comerciales de Qt desde marzo de 2011.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales [29].

3.2. Técnicas

3.2.1. Descriptor de características

Los descriptores visuales son herramientas muy necesarias para extraer la información que se considera importante de la imagen para que los clasificadores puedan realizar el aprendizaje. En este caso utilizamos los descriptores HOG y Haar que son dos de las características más utilizadas. Los HOG son histogramas de gradientes orientados que nos codifican la información que contienen las imágenes. En el segundo caso, los descriptores Haar se utilizan como derivadas discretas sobre imágenes y se usan para ser aprendidos por las cascadas mediante el clasificador Adaboost.

3.2.2. Clasificadores

Un clasificador es un sistema capaz de proporcionar una predicción de pertenencia a una clase como salida a partir de un conjunto de características tomadas como entradas. Un ejemplo de un clasificador es aquel que acepta datos de sueldos de una persona, edad, estado civil, dirección e historial de crédito y clasifica a la persona como aceptables o inaceptables para recibir una nueva tarjeta de crédito o préstamo.

Una de las posibles combinaciones de un conjunto de clasificadores es la cascada y ésta dará un mejor resultado dependiendo de la cantidad de datos que se introduzcan, el número de clasificadores, etc. La motivación para hacer uso de cascadas se debe a que éstas se usan para aprender conjunto desbalanceados de datos. Por ejemplo, en el caso de querer aprender cabeza contra cualquier otro elemento del mundo visual, incluiremos en la cascada más imágenes de lo que no es cabeza. [30]

3.2.3. Detección de contornos.

La detección de contornos tiene como objetivo recibir como entrada una imagen previamente binarizada y devolver como resultado un conjunto de contornos cerrados, los cuales están representados por polígonos. Para este propósito un operador de detección de contornos es aplicado a la imagen de entrada. Luego los contornos resultantes inicialmente analizados mediante una aproximación poligonal para encontrar únicamente los contornos cerrados. [19]

3.3. Librerías

3.3.1. Opencv



Figura 32. OpenCV, última versión 3.0

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas [31].

OpenCV es liberado bajo una licencia BSD y por lo tanto es gratis, tanto para uso académico y comercial. Cuenta con interfaces de C ++, C, Python y Java y es compatible con Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones de tiempo real. Escrito en optimizado C / C ++, la biblioteca puede tomar ventaja de procesamiento multi -core. Habilitado con OpenCL, se puede aprovechar la aceleración de hardware de la plataforma de computación heterogénea subyacente. Adoptado en todo el mundo, OpenCV tiene más de 47 mil personas de la comunidad de usuarios y el número estimado de descargas superiores a 9 millones. Rangos de uso del arte interactivo, a la inspección de minas, mapas de costura en la web o a través de la robótica avanzada [32].

3.3.2. CuteReports

Es una librería que permite diseñar e imprimir reportes en C++, se requiere de ciertos conocimientos del entorno de Linux para poder configurarla.

Permite combinar varios informes en un archivo XML. EL proyecto consta de dos partes: la biblioteca CuteReport y el diseñador de informes CuteReportDesigner.

3.3.3. Librerías multimedia

Son un conjunto de librerías que deben ser previamente configuradas e instaladas en el sistema operativo, para que la librería de Opencv pueda funcionar correctamente.

Entre las más importantes se menciona:

- Ffmpeg
- Libx264
- Libgtk
- Libjpeg
- V4l
- Xine-lib

e. Materiales y Métodos

Los materiales y métodos empleados en el desarrollo del presente proyecto de fin de carrera se detallan a continuación:

1. Materiales

Para que el desarrollo del presente Trabajo de Fin de Carrera sea exitoso fue necesario realizar una planificación y control en términos económicos de un conjunto de materiales, los cuales están ligados con las actividades que se desarrollaron en el proyecto.

A continuación se describen los materiales utilizados en el desarrollo del proyecto con sus respectivos costos:

1.1. Talento Humano

El presente proyecto se planificó para ser desarrollado por dos investigadores, contando con la asesoría de un docente de la carrera, para el desarrollo del proyecto se estimó un tiempo de 10 meses en el cual se trabajaron 5 horas diarias en días laborables, es decir, se trabajaron cien horas mensuales.

En la siguiente tabla se detallan el total de horas que se emplearon en el desarrollo del proyecto con sus respectivos costos:

TABLA 4. TALENTO HUMANO

Equipo de trabajo	Tiempo (Horas)	(\$) Precio / Hora	(\$) Valor Total
Investigadores	1000	4.00	4000.00
Director del PFC	400	0.00	0.00
SUBTOTAL			4000.00

1.2. Servicios

Además de contar con el talento humano, también se utilizaron servicios básicos que fueron necesarios para el desarrollo del proyecto.

Tal es el caso del uso de Internet y el transporte. El Internet fue utilizado como un medio para realizar investigación sobre el proyecto y el transporte para poder asistir a las tutorías planificadas por el Director del Proyecto de Fin de Carrera.

En la siguiente tabla se detallan estos servicios básicos.

TABLA 5. SERVICIOS

Servicio	Descripción	(\$) P. Unitario	(\$) Total
Internet	10 meses	20.05	200.50
Transporte	200 días	1.00	200.00
SUBTOTAL			400.00

1.3. Recursos Hardware y Software

Durante el desarrollo del Sistema se utilizaron recursos Hardware y Software. Entre los recursos Hardware están: una computadora, una impresora, un pendrive y una cámara, y entre los recursos Software está el paquete de Microsoft office 2013, las librerías que posee Opencv para Visión Artificial y algunas otras herramientas que nos facilitaron realizar toda la documentación del proyecto.

TABLA 6. HARDWARE Y SOFTWARE

RECURSOS HARDWARE				
	(\$)P. Unitario	T. vida (años)	T. uso (mes)	(\$) Depreciación
PC. Dell	1300.00	5	10	217.00
Impresora	250.00	5	10	42.00
Pendrive (8GB)	18.00	3	2	12.00
Cámara digital	150.00	10	10	70.00
Webcam	20	2	10	8.00
Cámara IP	145	4	10	30.00
Celular HTC	650	5	10	108.00
RECURSOS SOFTWARE				
	Descripción			(\$) Total
Windows 8.1	Para uso de Office 2013			45.00

Office 2013	Para la documentación y presentación.	75.00
Opencv	Para el desarrollo del Sistema.	0.00
Latex	Para el anteproyecto.	0.00
Ubuntu	Para desarrollo del programa.	0.00
SUBTOTAL		607.00

1.4. Materiales de Oficina

Para el desarrollo del proyecto también se emplearon algunos materiales de oficina, en los que se destacan insumos de papelería, anillados, empastados, etc...

En la siguiente tabla se detallan cada uno de estos materiales:

TABLA 7. MATERIALES DE OFICINA

	Cantidad	(\$ P. Unitario	(\$ Total
Insumos de papelería	1000	0.005	5.00
Anillados	7	2.60	18.20
Empastados	4	40.00	160.00
SUBTOTAL			183.20

El presupuesto total empleado para el desarrollo del presente Proyecto de Fin de Carrera se detalla a continuación, asumiendo un presupuesto para los imprevistos, el mismo que corresponde al 10% del total del presupuesto.

TABLA 8. PRESUPUESTO FINAL

	(\$ Total
Talento Humano	4000.00
Servicios	400.00
Recursos Hardware y Software	607.00
Materiales de Oficina	183.20
SUBTOTAL	5190.20
Imprevistos (10%)	519.02
Total PFC	5709.22

2. Métodos

Para el desarrollo del presente proyecto se utilizó una metodología de secuencia de pasos ordenados (modelo en cascada), los cuales sirvieron de guía a lo largo de la ejecución de diferentes actividades, alcanzando de esta manera los resultados esperados.

En la recolección y organización de la información obtenida, para sustentar el presente proyecto de tesis se hizo uso de los siguientes métodos y técnicas:

2.1. Método Deductivo

La aplicación de este método ha permitido conocer de manera específica cada uno de los problemas más comunes, partiendo del problema general de investigación que es: ¿El congestionamiento de vehículos en intersecciones controladas por semáforos con temporizadores provoca la intervención de un agente de tránsito, pérdida de tiempo, retrasos y posibles accidentes?, para poder localizar los problemas específicos y que se presentan generalmente en las principales calles de la ciudad.

2.2. Método Inductivo

Este método nos da la posibilidad de que a partir del análisis de cada uno de los problemas identificados que se ven presentes en las calles principales de la ciudad de Loja, como son accidentes de tránsito, desconocimiento de las leyes de tránsito entre otros nos permiten formular llegar al problema mencionados en el método anterior. Y así enfocarse directamente en resolver dicho problema.

2.3. Modelo en cascada

El modelo de ciclo de vida clásico, también denominado “modelo en cascada”, se basa en intentar hacer las cosas bien desde el principio, de una vez y para siempre. Se pasa, en orden, de una etapa a la siguiente sólo tras finalizar con éxito las tareas de verificación y validación propias de la etapa. Si resulta necesario, únicamente se da marcha atrás hasta la fase inmediatamente anterior.

Este modelo tradicional de ciclo de vida exige una aproximación secuencial al proceso de desarrollo del software. Por desgracia, esta aproximación presenta una serie de graves inconvenientes, entre los que cabe destacar:

- Los proyectos reales raramente siguen el flujo secuencial de actividades que propone este modelo.
- Normalmente, es difícil para el cliente establecer explícitamente todos los requisitos al comienzo del proyecto (entre otras cosas, porque hasta que no vea evolucionar el proyecto no tendrá una idea clara de qué es lo que realmente quiere).
- No habrá disponible una versión operativa del sistema hasta llegar a las etapas [33].

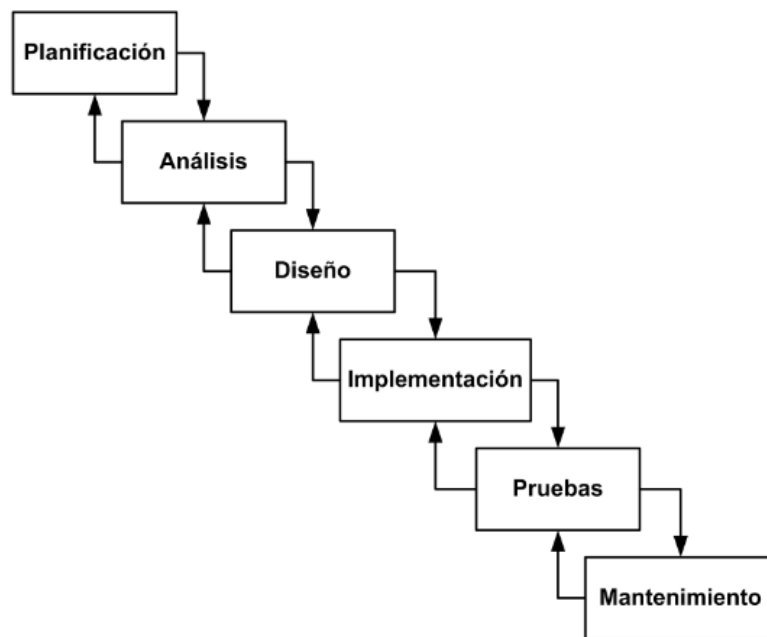


Figura 33. Metodología de Desarrollo del ciclo de vida clásico

3. Técnicas

Para la obtención de la información en el desarrollo del presente proyecto de investigación se utilizaron las siguientes técnicas:

- **Entrevista**

Esta técnica permitió obtener la información necesaria en cuanto a la manera de llevar a cabo cada uno de los procesos para el desarrollo de esta investigación, las entrevistas aplicadas a las personas expertas en cuanto a materia de tránsito se refiere permitió establecer los parámetros de entrada y las condiciones que debían tener los algoritmos desarrollados en el sistema, además de los conocimientos impartidos por el docente tutor.

- **Observación directa**

A través de esta técnica se logró observar y evidenciar las necesidades que se presentan en las principales avenidas de la ciudad y problemas que se generan diariamente.

- **Lectura comprensiva**

Permitió obtener un conocimiento ordenado y sistemático de los hechos e ideas relacionadas con el tema objeto de estudio, además sirvió para comprender cada una de las técnicas de visión artificial y los algoritmos que ayudaron a llevar a cabo el sistema final.

f. Resultados

1. Análisis de Requerimientos

En esta fase se realizó un análisis acerca de la perspectiva en la cual van a funcionar los algoritmos, en nuestro caso desde la perspectiva de un semáforo.

Para que la recolección de información se utilizó ciertas técnicas de recolección de datos, como son:

- La observación directa: que nos permitió conocer la forma en que transitan los vehículos y el área que debía ser limitada para el análisis vehicular.
- La entrevista: por parte del tutor de tesis que nos permitió conocer más a fondo la magnitud y el alcance del proyecto de semáforos inteligentes.

1.1. Requerimientos Funcionales

El Sistema permitirá:

TABLA 9. REQUERIMIENTOS FUNCIONALES DEL SISTEMA

REF.	Descripción	Categoría	Técnicas de recolección
REF001	Captar imágenes y videos a través de una cámara.	Evidente	Observación
REF002	Captar videos previamente grabados.	Evidente	Observación
REF003	Detectar vehículos a través de un clasificador.	Evidente	Observación
REF004	Detectar personas a través de un clasificador.	Evidente	Observación
REF005	Delimitar el área en la cual se va a realizar la detección.	Evidente	Observación
REF006	Utilizar un temporizador para controlar los eventos que se den.	Evidente	Observación

REF007	Almacenar los eventos generados en una base de datos.	Oculto	Observación
REF008	Generar un reporte de los posibles congestionamientos generados.	Evidente	Entrevista
REF009	Generar un reporte de los posibles accidentes generados.	Evidente	Entrevista

1.2. Requerimientos no Funcionales

De la misma forma que los requerimientos funcionales, también existen los requerimientos no funcionales, en donde se dice que el sistema deberá:

TABLA 10. REQUERIMIENTOS NO FUNCIONALES DEL SISTEMA.

REF.	Descripción	Categoría	Técnicas de recolección
REF001	Poseer una interfaz que sea amigable para el usuario.	Evidente	Entrevista
REF002	Facilitar su uso para poder realizar las pruebas apropiadas.	Evidente	Entrevista

2. Diseño

2.1. Algoritmo para la detección de vehículos y posible congestionamiento

A continuación se muestra el algoritmo que permite realizar la detección de los vehículos en tiempo real. El algoritmo está estructurado en un diagrama de flujo. Lo que hace el algoritmo al principio es una evaluación sobre si existe o no un vehículo. Seguidamente se analiza si existe un determinado número de vehículos, en caso de ser afirmativo se inicia un contador por un determinado tiempo si ambas condiciones se dan se presenta una alerta.

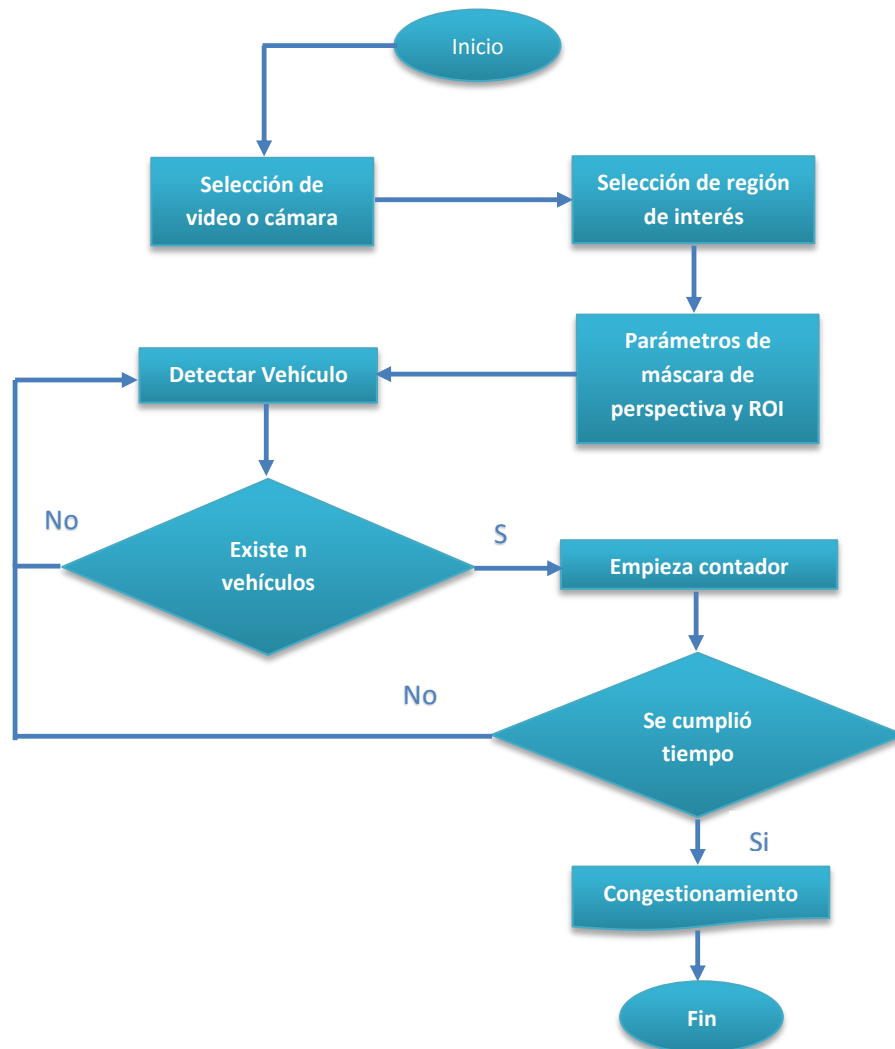


Diagrama 1. Diagrama de flujo del algoritmo para la detección de congestionamientos

2.2. Algoritmo para la detección de vehículos y posible accidente

El algoritmo realiza una evaluación sobre si existe o no un vehículo. Seguidamente se analiza si existe uno o más vehículos estáticos, si es afirmativo, se detecta la presencia de personas, en caso de ser afirmativo se inicia un contador por un determinado tiempo, si estas condiciones se dan se presenta una alerta de posible accidente.

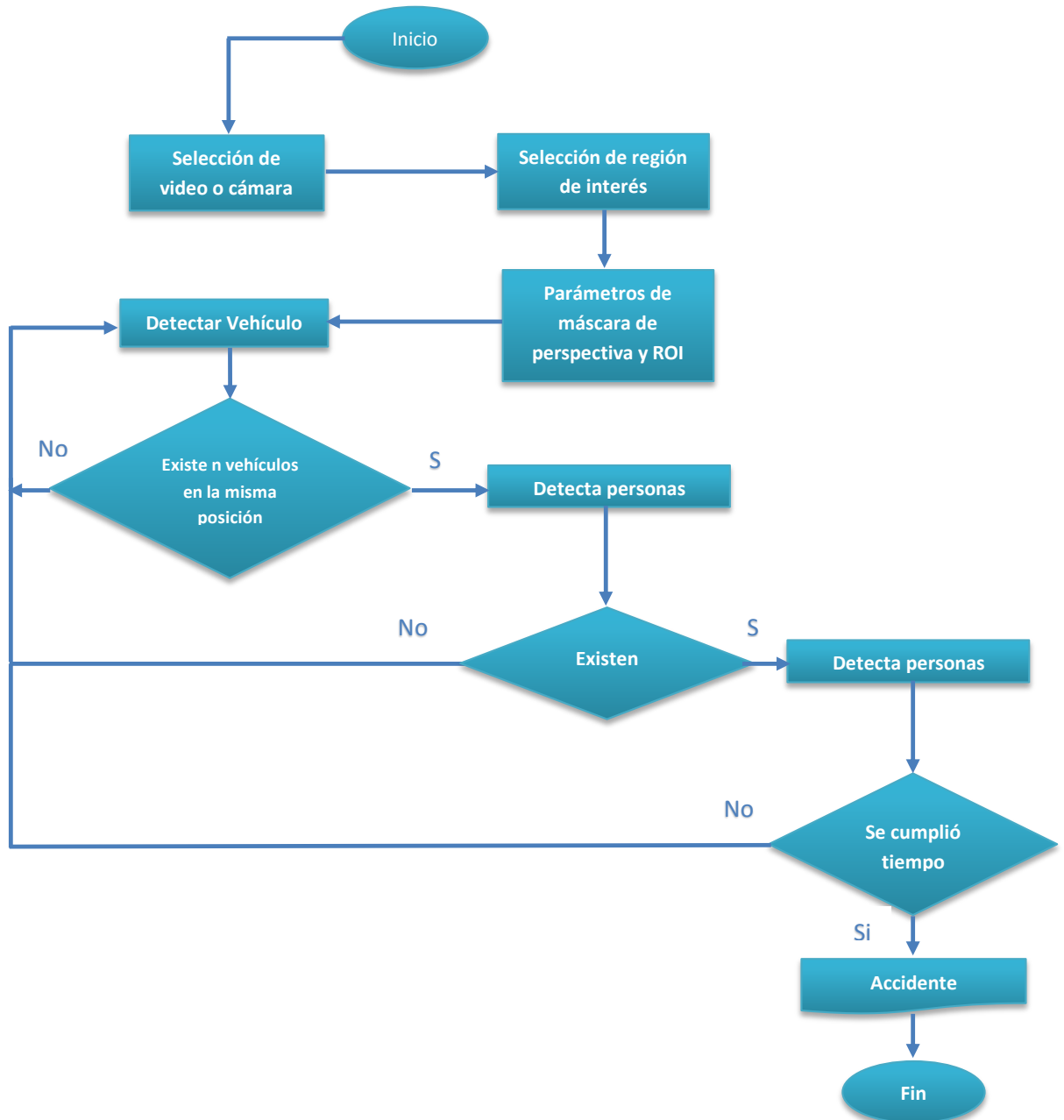


Diagrama 2. Diagrama de flujo del algoritmo para la detección de accidentes

2.3. Prototipado

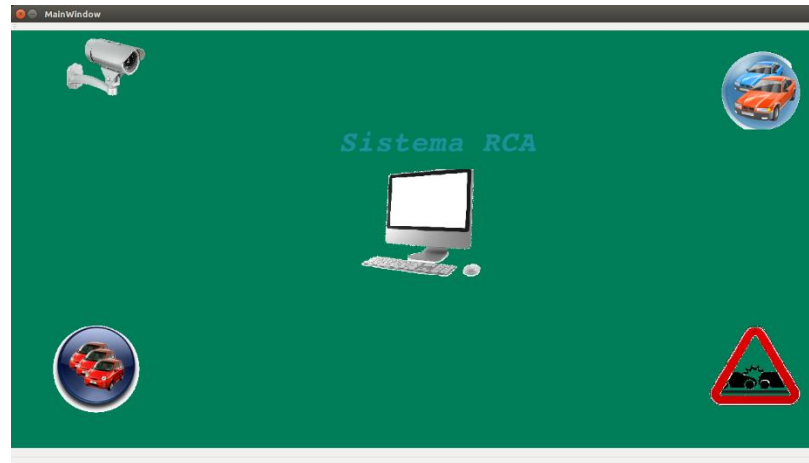


Figura 34. Pantalla principal del sistema



Figura 35. Pantalla detección posibles accidentes



Figura 36. Pantalla detección posibles congestionamientos

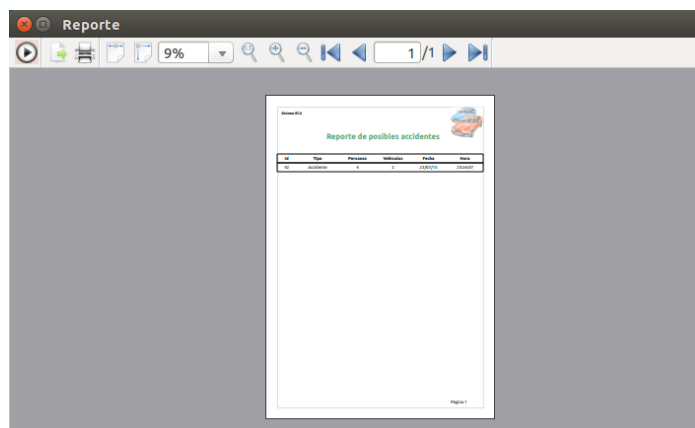


Figura 37. Pantalla generación de reportes de eventos de accidentes

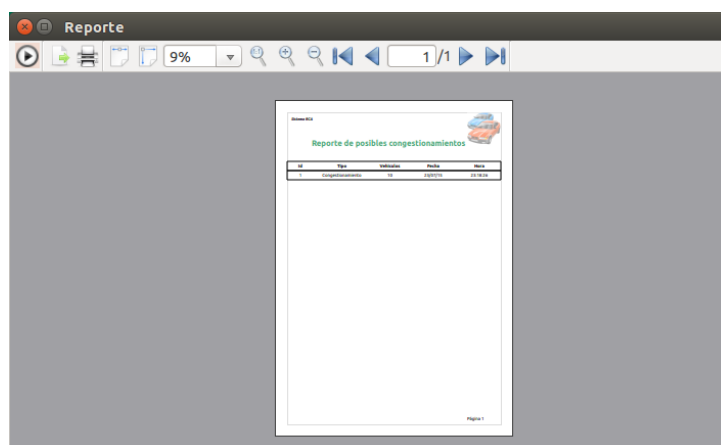
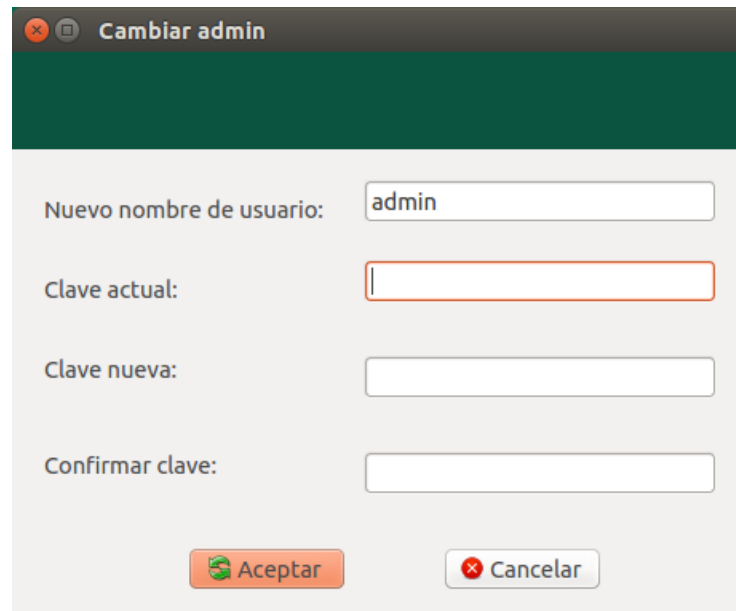


Figura 38. Pantalla generación de reportes de eventos de congestionamientos



The screenshot shows a window titled "Cambiar admin" with a dark green header. Below the header, there are four input fields: "Nuevo nombre de usuario:" with the value "admin", "Clave actual:" (empty), "Clave nueva:" (empty), and "Confirmar clave:" (empty). At the bottom, there are two buttons: "Aceptar" (green) and "Cancelar" (red).

Figura 39. Pantalla generación de reportes de eventos de congestionamientos

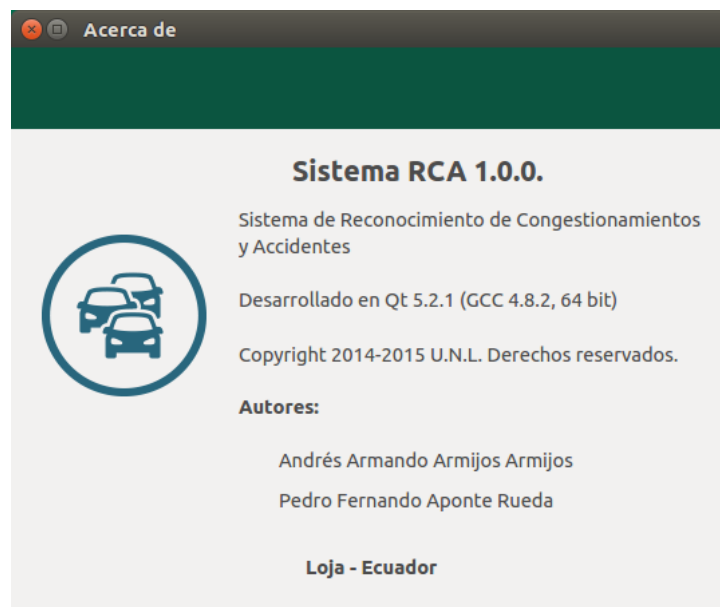


Figura 40. Pantalla Acerca del programa

3. Implementación

3.1. Construcción del clasificador en cascada

Como se mencionó anteriormente para el desarrollo del algoritmo correspondiente a la detección de vehículos desde la perspectiva de un semáforo fue necesario construir un clasificador en cascada. Para realizar la detección de vehículos se construyó el clasificador en la plataforma Windows 8.1, debido a que se emplearon gran cantidad de imágenes (muestras).

3.2. Clasificador en cascada para la detección de vehículos

Ver Anexo 2

3.3. Implementación de los algoritmos

En la implementación de los algoritmos y la codificación del Sistema se utilizó el sistema Operativo Ubuntu 14.04, el Lenguaje de programación C++, el IDE QT Creator y SQLite para la Base de Datos.

3.4. Codificación del algoritmo para la detección de vehículos, posible congestionamiento y posible accidente.

En esta sección se presenta la codificación del algoritmo en Qt con C++.

3.4.1. Librerías y cabeceras

Lo primero que se hace es incluir las librerías y cabeceras.

TABLA 11. LIBRERÍAS Y CABECERAS.

```
#include "accidentes.h"  
#include "ui_accidentes.h"  
#include <QMainWindow>  
#include <QMessageBox>  
#include "QTimer"  
#include <QApplication>  
#include <iostream>  
#include <sstream>
```

```

#include <QThread>
#include <QFileDialog>
#include "opencv/cxcore.h"
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv/cv.h"
#include "opencv2/opencv.hpp"
#include "opencv2/video/video.hpp"
#include "opencv2/ml/ml.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <QDebug>
#include <opencv/cv.h>
#include <opencv/highgui.h>
#include <iostream>
#include <QtCore/qmath.h>
#include <camaraac.h>

```

3.4.2. Función para ingresar los puntos a través del Ratón (Mouse).

La función MousePosition sirve para ingresar los 4 puntos que delimitan el área de interés en la cual detectará el clasificador los objetos o vehículos.

TABLA 12. FUNCIÓN PARA INGRESAR LOS PUNTOS A TRAVÉS DEL RATÓN.

```

void Accidentes::Mouse_Position()
{
    int x = ui->lblCamaraA->x;
    int y = ui->lblCamaraA->y;
    puntosA++;
    switch(puntosA){
        case 1: pt1A.x=x;
                pt1A.y=y;
                break;
        case 2: pt2A.x=x;
                pt2A.y=y;
                break;
        case 3: pt3A.x=x;
                pt3A.y=y;
                break;
        case 4: pt4A.x=x;
                pt4A.y=y;
                break;
    }
    if(puntosA==4){
        delimitar_area();
    }
}

```

3.4.3. Área de interés delimitada

Aquí lo que se hace es delimitar el área de interés en base a los 4 puntos ingresados con el mouse. En cada frame se recorta solo el área de interés para eliminar las zonas innecesarias en el momento que se hace la detección de los objetos.

TABLA 13. FUNCIÓN PARA DELIMITAR EL ÁREA DE INTERÉS.

```
void Accidentes::delimitar_area()
{
    maskkA.create(dA.rows, dA.cols, CV_8UC1);
    for(int i=0; i<maskkA.cols; i++)
        for(int j=0; j<maskkA.rows; j++)
            maskkA.at<uchar>(Point(i,j)) = 0;
    vector<Point> ROI_Poly;
    vector<Point> ROI_Vertices;
    ROI_Vertices.push_back(pt1A);
    ROI_Vertices.push_back(pt2A);
    ROI_Vertices.push_back(pt3A);
    ROI_Vertices.push_back(pt4A);
    approxPolyDP(ROI_Vertices, ROI_Poly, 1.0, true);
    fillConvexPoly(maskkA, &ROI_Poly[0], ROI_Poly.size(), 255, 8, 0);
    imageDestA.create(dA.rows, dA.cols, CV_8UC3);
    imageDestAer.create(dA.rows, dA.cols, CV_8UC3);
    dA.copyTo(imageDestA, maskkA);
    clasificadorA.load("/home/pedro/Escritorio/output.xml");
    clasificadorB.load("/home/pedro/QTCreator/P1/personas.xml");
    imageDestAer.create(dA.rows, dA.cols, CV_8UC3);
    clikeadasA++;
    tiempoA->start(40);
    segundos = 0;
    tiempoAEvento->start(1000);
}
```

Para que se tenga una mejor idea de lo expuesto, en la imagen siguiente se ilustra el área en la cual solo se filtran los objetos que están en movimiento y en la otra el área de interés trazada en el frame final presentado.

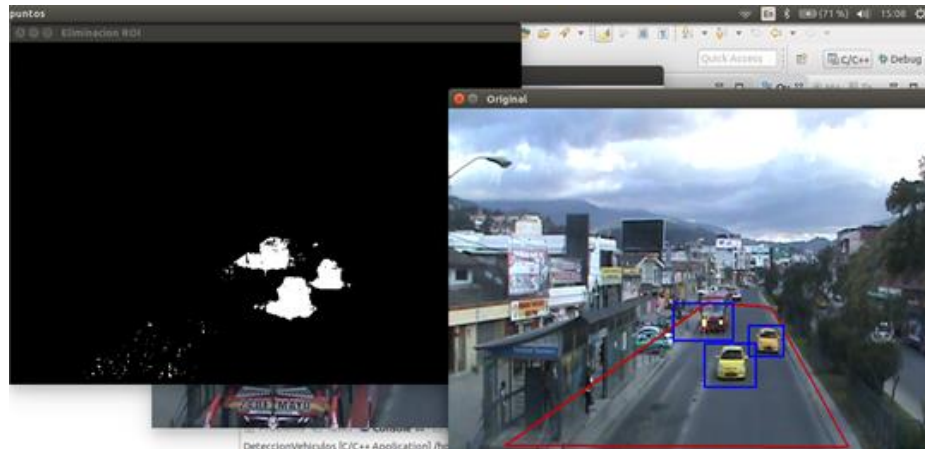


Figura 41. Área de interés trazada

3.4.4. Implementación del clasificador en cascada

Luego de haber construido el clasificador en cascada para la detección de vehículos, el siguiente paso consiste en implementarlo.

En la siguiente figura se puede observar la implementación.

TABLA 14. FUNCIÓN DONDE SE IMPLEMENTA LOS CLASIFICADORES.

```
void Accidentes::capturar()
{
    cA.read(aA);
    aA.copyTo(imageDestAer, maskkA);
    cv::line(aA,pt1A,pt2A,cv::Scalar(0,255,0), 2, 4);
    cv::line(aA,pt2A,pt3A,cv::Scalar(0,255,0), 2, 4);
    cv::line(aA,pt3A,pt4A,cv::Scalar(0,255,0), 2, 4);
    cv::line(aA,pt4A,pt1A,cv::Scalar(0,255,0), 2, 4);
    clasificadorA.detectMultiScale(imageDestAer,reconocidosA,1.2,3,0,cvSize(30,30));
    clasificadorB.detectMultiScale(imageDestAer,reconocidosB,1.2,2,0,cvSize(60,150),cvSize(84,210));
    for (size_t var = 0; var < reconocidosA.size(); ++var) {
        rectangle(aA,Point(reconocidosA[var].x,reconocidosA[var].y),
        Point(reconocidosA[var].x + reconocidosA[var].width, reconocidosA[var].y +
        reconocidosA[var].height),cvScalar(255,0,0));
        putText(aA, "Vehiculo", cvPoint(reconocidosA[var].x,reconocidosA[var].y),
        CV_FONT_NORMAL, 0.5, cvScalar(170,0,0),1,8);
    }
    for (size_t var = 0; var < reconocidosB.size(); ++var) {
        rectangle(aA,Point(reconocidosB[var].x,reconocidosB[var].y),
```

```

Point(reconocidosB[var].x + reconocidosB[var].width, reconocidosB[var].y +
reconocidosB[var].height),cvScalar(0,0,255));
putText(aA, "Persona", cvPoint(reconocidosB[var].x,reconocidosB[var].y),
CV_FONT_NORMAL, 0.5, cvScalar(170,0,0),1,8);
}
if(reconocidosA.size()==totalVehiculosA){
    if(reconocidosB.size()>0){
        if(distanciaActual==distanciaAnterior){
            accidente = true;
        }
    }
}
else{
    accidente = false;
}
}
ui->lcdNumberAV->display(2);
ui->lcdNumberP->display((int)reconocidosB.size());
QImage framePQimage=transformar(aA);
ui->lblCamaraA->setAlignment(Qt::AlignCenter);
ui->lblCamaraA->setPixmap(QPixmap::fromImage(framePQimage));
}

```

3.4.5. Temporizador para congestionamiento

Esta función lo que hace es empezar un contador que en este caso lo hemos fijado en 60, o sea que mientras se cumpla la condición de que existe determinado número de vehículos por 60 segundos, se toma en cuenta como congestionamiento.

TABLA 15. FUNCIÓN TEMPORIZADOR CONGESTIONAMIENTO.

```

void Congestionamiento::temporizadorCongestionamiento()
{
    if(congestionamiento){
        segundosC++;
        ui->lcdNumberC->display(segundosC);
        if(segundosC>segundosI){
            QPixmap image("/home/pedro/QTCreator/P1/iconos/congest.png");
            ui->lblCongestionamiento->setText("Congestionamiento");
            residuo = segundosC % 2;
            if(residuo == 0){
                ui->lblImagenC->setPixmap(image);
            }else{
                ui->lblImagenC->setPixmap(NULL);
            }
            insertarDatosEvento("Congestionamiento");
        }
    }
    else{
        segundosC = 0;
        ui->lcdNumberC->display(segundosC);
    }
}

```

3.4.6. Temporizador para accidente

Esta función contabiliza un minuto siempre y cuando se cumpla la condición de que 1 o más vehículos no se han desplazado de sus respectivas coordenadas y se detecte la presencia de personas, para lo cual se utiliza dos vectores con las coordenadas de los objetos detectados en cada frame y se realiza una comparación de los valores de las coordenadas, si no se detecta ningún desplazamiento durante 60 segundos se considera que existe accidente. Cabe señalar que el tiempo a considerar puede ser modificado, por motivos de ilustración y explicación se tomó 60 segundos.

TABLA 16. FUNCIÓN TEMPORIZADOR ACCIDENTE.

```
void Accidentes::temporizadorAccidente()
{
    if(accidente){
        segundos++;
        ui->lcdNumberA->display(segundos);
        if(segundos>segundosIA){
            QPixmap image("/home/pedro/QTCreator/P1/iconos/zone_2.jpg");
            ui->lblAccidente->setText("Accidente");
            residuoA = segundos % 2;
            if(residuoA == 0){
                ui->lblImagenA->setPixmap(image);
            }else{
                ui->lblImagenA->setPixmap(NULL);
            }
            insertarDatosEvento("Accidente");
        }
    }else{
        segundos = 0;
        ui->lcdNumberA->display(segundos);
    }
}
```

3.4.7. Comparador de desplazamiento o distancia

Esta función compara las coordenadas de dos vehículos, en caso de que no se detecte desplazamiento o no se alteren el valor de las coordenadas, se puede tomar como punto de referencia para establecer que los vehículos no presentan desplazamiento alguno.

TABLA 17. FUNCIÓN DISTANCIA O DESPLAZAMIENTO DE LOS VEHÍCULOS.

```
double Accidentes::distancia(Point punto1, Point punto2)
{
    double xs = (punto2.x - punto1.x)*(punto2.x - punto1.x);
    double ys = (punto2.y - punto1.y)*(punto2.y - punto1.y);
    return qSqrt(xs+ys);
}
```

3.4.8. Función para insertar los eventos en la base de datos.

Esta función permite insertar un registro del evento que se suscita en la base de datos. Sirve tanto para registrar el evento de posible congestionamiento, como para posible accidente.

TABLA 18. FUNCIÓN PARA INSERTAR LOS EVENTOS EN LA BASE DE DATOS.

```
void Accidentes::insertarDatosEvento(QString tipo){
    QString consulta;
    consulta.append("INSERT INTO evento("
        "tipo,"
        "fecha,"
        "hora,"
        "personas,"
        "vehiculos)"
        "VALUES("
        """+tipo+""","
        """+obtenerFechaAS()+""","
        """+obtenerHoraAS()+""","");
    consulta.append(QString::number(personas));
    consulta.append(",");
    consulta.append(QString::number(vehiculos));
    consulta.append(");");
    QSqlQuery insertar;
    insertar.prepare(consulta);
    if(insertar.exec()){
        qDebug()<<"El evento se ha insertado correctamente";
    }else{
        qDebug()<<"El evento no se ha insertado correctamente";
        qDebug()<<"ERROR" <<insertar.lastError();
    }
}
```

4. Pruebas

Luego de haber construido e implementado el clasificador en los algoritmos correspondientes, pasamos a comprobar la precisión de los mismos.

4.1. Detección de vehículos

Para la construcción del clasificador en cascada para la detección de vehículos se emplearon 3000 imágenes positivas y 1500 imágenes negativas, obteniendo resultados muy efectivos, se obtuvieron los siguientes resultados:

4.1.1. Pruebas con imágenes

TABLA 19. DETECCIÓN VEHÍCULOS EN LLUVIA

Condición del día:		Lluvia
Lugar:	Paso peatonal Terminal Terrestre	
Hora:	08:30 A.M.	
	Número	%
Vehículos reales	89	100
Detectados	78	87,6
Falsos positivos	5	5,6
Falsos negativos	6	6,8

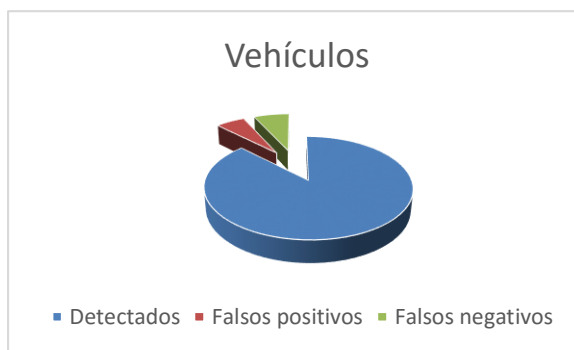


Diagrama 3. Vehículos detectados en lluvia

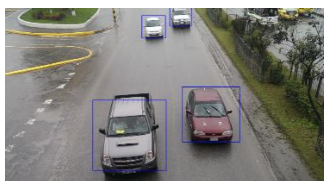


Figura 42. Ejemplo detección en lluvia

TABLA 20. DETECCIÓN DE VEHÍCULOS CON LLOVIZNA Y SOL

Condición del día:		Llovizna con sol	
Lugar:		Paso peatonal Terminal Terrestre	
Hora:		09:30 A.M.	
		Número	%
Vehículos reales		43	100
Detectados		41	95,4
Falsos positivos		1	2,3
Falsos negativos		1	2,3

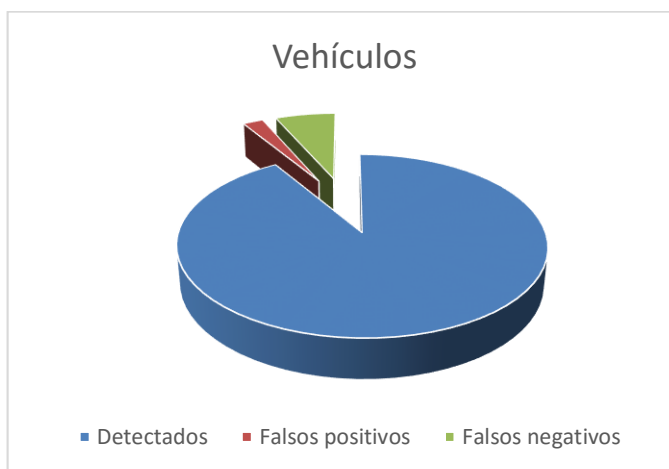


Diagrama 4. Detección de vehículos con llovizna y sol

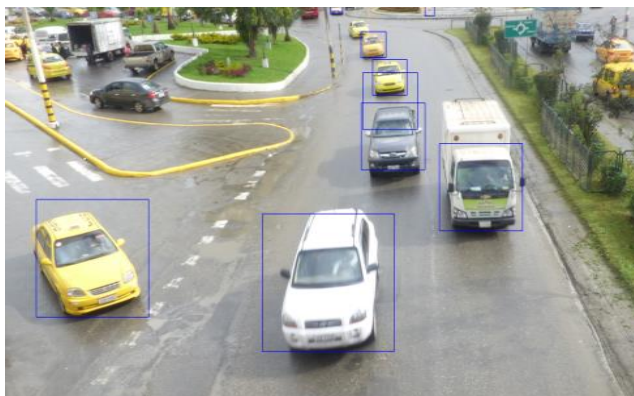


Figura 43. Ejemplo de detección en llovizna con sol

TABLA 21. DETECCIÓN DE VEHÍCULOS EN DÍA CON SOL

Condición del día:		Soleado	
Lugar:	Paso peatonal Terminal Terrestre		
Hora:	09:30 A.M.		
	Número	%	
Vehículos reales	209	100	
Detectados	201	96,2	
Falsos positivos	4	1,9	
Falsos negativos	4	1,9	

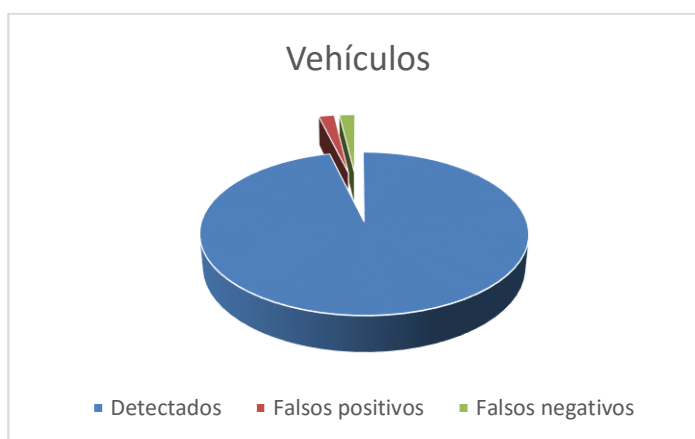


Diagrama 5. Detección de vehículos en día con sol

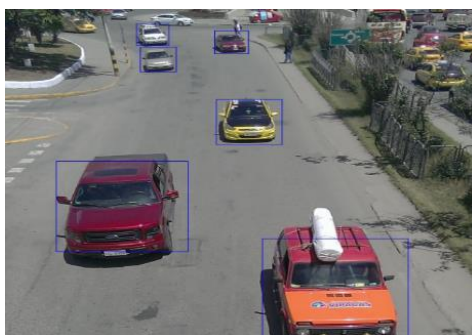


Figura 44. Ejemplo de detección en día con sol

TABLA 22. DETECCIÓN DE VEHÍCULOS EN DÍA SOMBREADO

Condición del día:		Sombreado	
Lugar:	Paso peatonal Terminal Terrestre		
Hora:	09:30 A.M.		
	Número	%	
Vehículos reales	162	100	
Detectados	147	90,7	
Falsos positivos	11	6,8	
Falsos negativos	4	2,5	

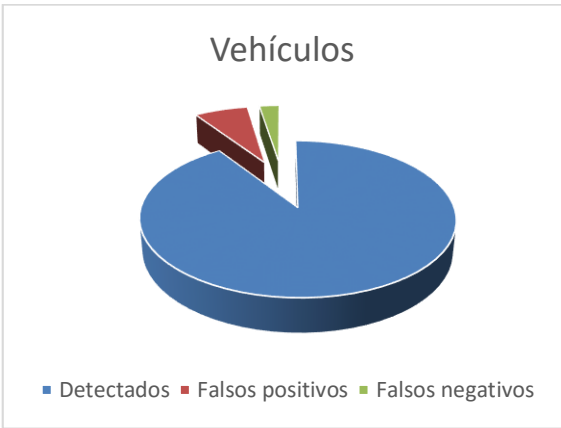


Diagrama 6. Detección de vehículos en día sombreado

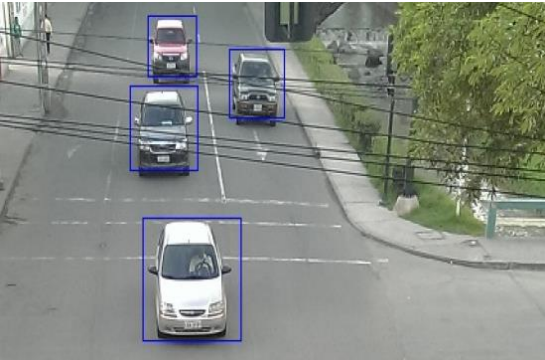


Figura 45. Ejemplo de detección de vehículos en día sombreado

4.1.2. Pruebas con video (cámara ip)

TABLA 23. DETECCIÓN DE VEHÍCULOS EN LLUVIA CON CÁMARA IP

Condición del día:	Lluvia	
Lugar:	Paso peatonal, Terminal Terrestre	
Hora:	09:30 P.M.	
	Número	%
Vehículos reales	50	100
Detectados	47	94,0
Falsos positivos	1	2,0
Falsos negativos	2	4,0

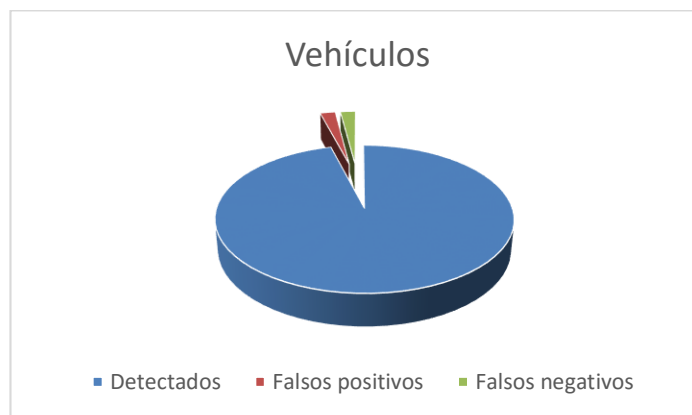


Diagrama 7. Detección de vehículos en lluvia con cámara ip

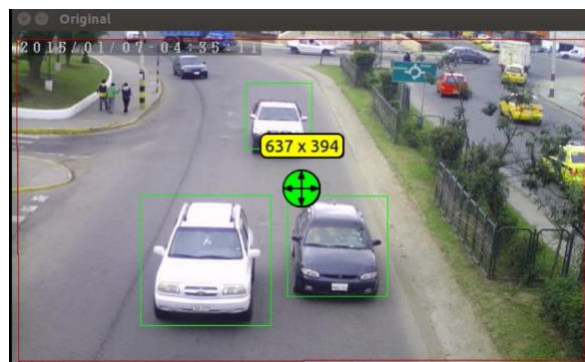


Figura 46. Detección a través de cámara ip, Terminal Terrestre

Tabla 24. DETECCIÓN VEHICULAR DÍA SOMBREADO, TERMINAL TERRESTRE

Condición del día:		Sombreado
Lugar:	Paso peatonal, Terminal Terrestre	
Hora:	04:27 P.M.	
	Número	%
Vehículos reales	75	100
Detectados	73	97,4
Falsos positivos	1	1,3
Falsos negativos	1	1,3

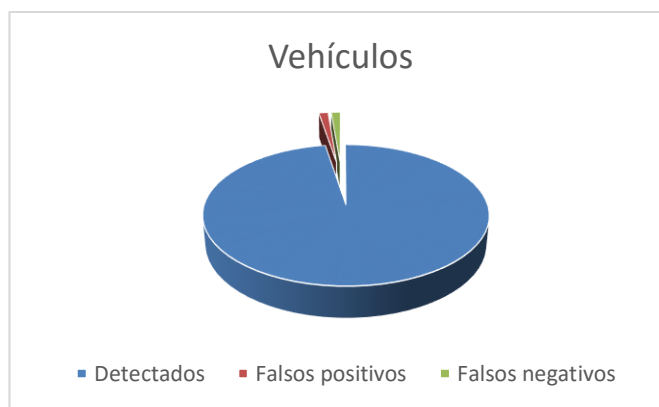


Diagrama 8. Detección de vehículos día sombreado, Terminal Terrestre



Figura 47. Detección a través de cámara ip

TABLA 25. DETECCIÓN DE VEHÍCULOS PASO PEATONAL MERCADILLO

Condición del día:	Sombreado	
Lugar:	Paso peatonal, Mercadillo y Avenida Universitaria	
Hora:	05:15 P.M.	
	Número	%
Vehículos reales	77	100
Detectados	75	97,4
Falsos positivos	1	1,3
Falsos negativos	1	1,3

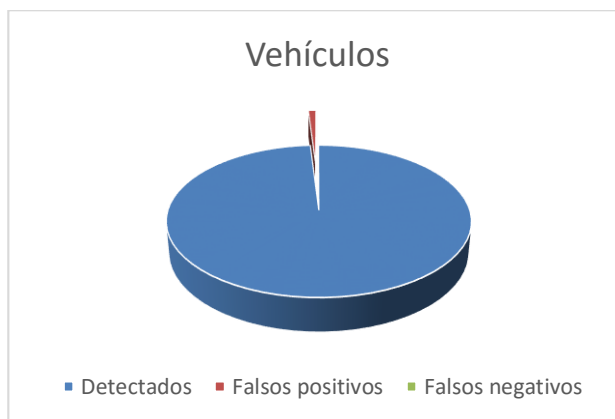


Diagrama 9. Detección de vehículos día sombreado, mercadillo sur-norte



Figura 48. Detección a través de cámara ip

4.2. Detección de personas

Para la construcción del clasificador en cascada para la detección de personas se emplearon 1000 imágenes positivas y 500 imágenes negativas, obteniendo los siguientes resultados:

4.2.1. Pruebas con imágenes

TABLA 26. DETECCIÓN DE PERSONAS PASO PEATONAL CALLE MERCADILLO

Condición del día:	Sombreado	
Lugar:	Paso peatonal Terminal Terrestre	
Hora:	08:30 A.M.	
	Número	%
Personas	50	100
Detectadas	45	90
Falsos positivos	2	4
Falsos negativos	3	6



Diagrama 10. Detección de personas paso peatonal calle Mercadillo

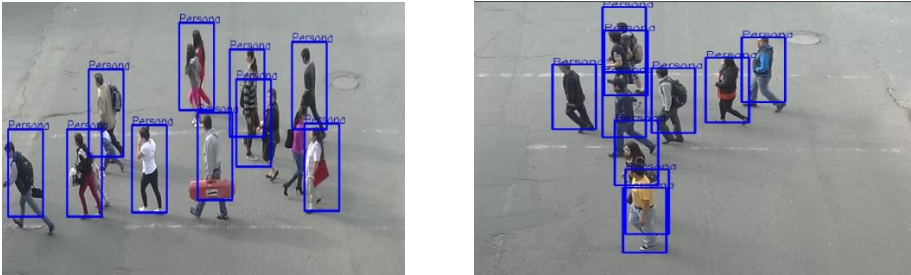


Figura 49. Ejemplo de detección de personas en calle Mercadillo

TABLA 27. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL

Condición del día:		Normal	
Lugar:	Estacionamiento Alameda Real		
Hora:	05:30 P.M.		
	Número	%	
Personas	20	100	
Detectadas	19	95	
Falsos positivos	1	5	
Falsos negativos	0	0	



Diagrama 11. Detección de personas estacionamiento Alameda Real



Figura 50. Ejemplo de detección de personas en estacionamiento

TABLA 28. DETECCIÓN DE PERSONAS IMÁGENES INTERNET

Condición del día:	Variable	
Lugar:	Imágenes de internet	
Hora:	-	
	Número	%
Personas	50	100
Detectadas	47	94
Falsos positivos	2	4
Falsos negativos	1	2



Diagrama 12. Detección de personas imágenes internet



Figura 51. Ejemplo de detección de personas con imágenes de internet

4.2.2. Pruebas con video (cámara IP)

TABLA 29. DETECCIÓN DE PERSONAS PASO PEATONAL CALLE MERCADILLO

Condición del día:		Sombreado
Lugar:	Paso peatonal Terminal Terrestre	
Hora:	08:30 A.M.	
	Número	%
Personas	30	100
Detectadas	28	93,3
Falsos positivos	2	6,7
Falsos negativos	3	10



Diagrama 13. Detección de personas calle Mercadillo

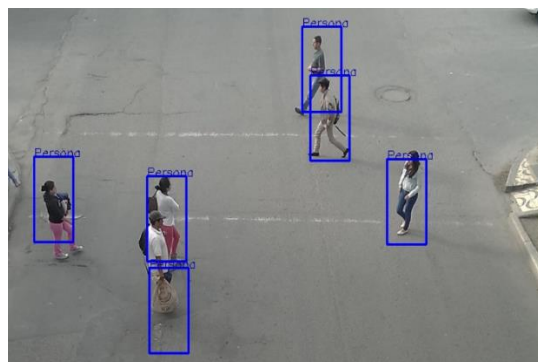


Figura 52. Detección de personas calle Mercadillo

TABLA 30. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL

Condición del día: Normal		
Lugar:	Estacionamiento Alameda Real	
Hora:	05:30 P.M.	
	Número	%
Personas	10	100
Detectadas	10	100
Falsos positivos	0	0
Falsos negativos	1	10



Diagrama 14. Detección de personas estacionamiento Alameda



Figura 53. Detección de personas en estacionamiento Alameda

4.3. Detección de vehículos y personas

Para la detección de vehículos y personas se integró los clasificadores de los mismos, obteniendo los siguientes resultados:

4.3.1. Pruebas con imágenes

TABLA 31. DETECCIÓN DE PERSONAS Y VEHÍCULOS CALLE MERCADILLO

Condición del día:		Sombreado	
Lugar:	Paso peatonal, calle Mercadillo		
Hora:	08:30 A.M.		
	Número	%	
Personas y vehículos	14	100	
Objetos detectados	14	100	
Falsos positivos	0	0	
Falsos negativos	0	0	



Diagrama 15. Detección de personas y vehículos paso peatonal calle Mercadillo



Figura 54. Detección de personas y vehículos paso peatonal calle Mercadillo

TABLA 32. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL

Condición del día:	Sombreado	
Lugar:	Estacionamiento Alameda Real	
Hora:	05:30 P.M.	
	Número	%
Personas y vehículos	7	100
Objetos detectados	7	100
Falsos positivos	0	0
Falsos negativos	0	0



Diagrama 16. Detección de personas y vehículos estacionamiento Alameda Real

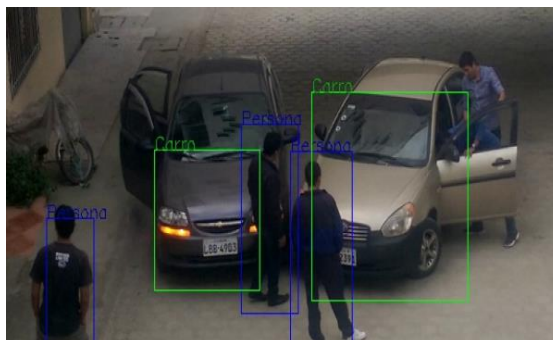


Figura 55. Detección de personas y vehículos estacionamiento Alameda Real

4.3.2. Pruebas con video (cámara IP)

TABLA 33. DETECCIÓN PERSONAS Y VEHÍCULOS CALLE MERCADILLO

Condición del día:	Sombreado	
Lugar:	Paso peatonal, calle Mercadillo	
Hora:	08:30 A.M.	
	Número	%
Personas y vehículos	8	100
Objetos detectados	8	100
Falsos positivos	0	0
Falsos negativos	2	20



Diagrama 17. Detección de personas y vehículos paso peatonal calle Mercadillo

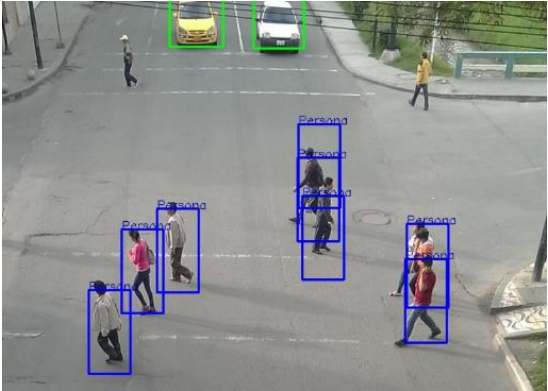


Figura 56. Detección de personas y vehículos calle Mercadillo

TABLA 34. DETECCIÓN DE PERSONAS ESTACIONAMIENTO ALAMEDA REAL

Condición del día:		Sombreado	
Lugar:	Estacionamiento Alameda Real		
Hora:	05:30 P.M.		
	Número	%	
Personas y vehículos	5	100	
Objetos detectados	0	100	
Falsos positivos	0	0	
Falsos negativos	1	20	



Diagrama 18. Detección de personas y vehículos estacionamiento Alameda Real

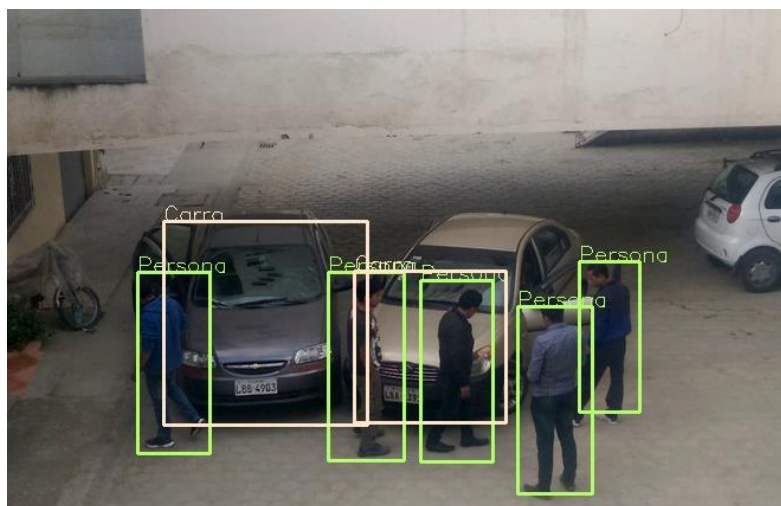


Figura 57. Detección de personas y vehículos estacionamiento Alameda

Balance de detección con imágenes

TABLA 35. DETECCIÓN DE VEHÍCULOS CON IMÁGENES

Condición del día	%Aciertos	%F.P.	%F.N.
Lluvia	87,6	5,6	6,8
Llovizna con sol	95,4	2,3	2,3
Sol	96,2	1,9	1,9
Sombra	90,7	6,8	2,5
Promedio	92,5	4,1	3,4

TABLA 36. DETECCIÓN DE PERSONAS CON IMÁGENES

Condición del día	%Aciertos	%F.P.	%F.N.
Sombreado	90	4	6
Normal	95	5	0
Variable	94	4	2
Promedio	93	4,3	2,7

TABLA 37. DETECCIÓN DE VEHÍCULOS Y PERSONAS CON IMÁGENES

Condición del día	%Aciertos	%F.P.	%F.N.
Sombreado	100	0	0
Sombreado	100	0	0
Promedio	100	0	0

En la mayoría de los balances o promedios de detección es sobre el 90%.

Balance de detección con video

TABLA 38. DETECCIÓN DE VEHÍCULOS EN TIEMPO REAL

Lugar	%Aciertos	%F.P.	%F.N.
Terminal Terrestre	94	2	4
Terminal Terrestre	97,4	1,3	1,3
Calle Mercadillo	97,4	1,3	1,3
Promedio	96,3	1,5	2,2

TABLA 39. DETECCIÓN DE PERSONAS EN TIEMPO REAL

Lugar	%Aciertos	%F.P.	%F.N.
Calle Mercadillo	93,3	6,7	10
Alameda Real	100	0	10
Promedio	96,65	3,35	10

TABLA 40. DETECCIÓN, VEHÍCULOS Y PERSONAS EN TIEMPO REAL

Lugar	%Aciertos	%F.P.	%F.N.
Calle Mercadillo	100	0	20
Alameda Real	100	0	20
Promedio	100	0	20

En la mayoría de los balances o promedios de detección es sobre el 90%.

4.4. Registro del evento de posible congestionamiento

Para realizar las pruebas se tomaron como referencia el número de vehículos y el tiempo a considerar para que se produzca el evento. Se hicieron las pruebas en 2 videos, se tomó en cuenta que si existen 10 o más vehículos, se considere como un posible congestionamiento. El tiempo que se tomó para el temporizador fue de 60 segundos. Cabe destacar que estos parámetros fueron tomados en base a las entrevistas realizadas.

Los resultados fueron los siguientes:

TABLA 41. PRUEBA CONGESTIONAMIENTO

Condición del día:		Soleado	
Lugar:	Paso peatonal Terminal Terrestre		
Hora:	01:00 P.M.		
	Número	%	
Veces que existieron más de 10 vehículos	2	100	
Posible congestionamiento detectado	2	100	
Falsos positivos	0	0	
Falsos negativos	0	0	



Diagrama 19. Prueba congestionamiento



Figura 58. Detección de posible congestionamiento.

Reporte

Sistema RCA

Reporte de posibles congestionamientos

Id	Tipo	Vehiculos	Fecha	Hora
1	Congestionamiento	10	23/07/15	23:18:26

Figura 59. Registro del posible congestionamiento.

4.5. Registro del evento de posible accidente

Para realizar esta prueba se utilizó 2 videos en los cuales se detecta si dos o más vehículos no se han desplazado de su posición actual y si se detecta presencia de personas sea 1 o más. El tiempo que se tomó para el temporizador fue de 30 segundos.

Lo que hace el algoritmo es verificar si uno o más vehículos permanecen estáticos en las mismas coordenadas, tal como se puede observar en la figura anterior, si el vehículo que posee la coordenada (420, 415) y el vehículo que posee la coordenada (432,390), no se mueven de la misma coordenada por un lapso de 60 segundos y además se detecta la presencia de personas en el área delimitada, entonces se envía a guardar en la base de datos una alerta de un posible accidente.

Los resultados fueron los siguientes:

TABLA 42. PRUEBA ACCIDENTE

Condición del día:		Sombreado	
Lugar:	Estacionamiento Alameda Real		
Hora:	05:30 P.M.		
	Número	%	
Accidente presente	2	100	
Posible accidente detectado	2	100	
Falsos positivos	0	0	
Falsos negativos	0	0	



Diagrama 20. Prueba accidente



Figura 60. Detección de posible accidente.

Reporte

Sistema RCA

Reporte de posibles accidentes

Id	Tipo	Personas	Vehículos	Fecha	Hora
42	Accidente	4	2	23/07/15	23:26:07

Figura 61. Registro de posible accidente

g. Discusión

Luego de haber concluido con el desarrollo del proyecto de fin de carrera titulado **“Desarrollo de un Sistema de Visión Artificial para la detección de accidentes y/o congestionamientos vehicular”** es momento de realizar un análisis acerca de los objetivos establecidos para el desarrollo del presente proyecto, con el fin de evaluar su cumplimiento, los mismos que a continuación se describen:

- ✓ Desarrollar un componente que permita la detección de accidentes a través de la Visión Artificial en tiempo real

Este objetivo se logró, primero al delimitar la región de interés, descartando zonas y posibles objetos erróneos, además por la detección que permite realizar el clasificador en cascada con imágenes tomadas desde la perspectiva de un semáforo (pasos peatonales ubicados a la altura de un semáforo). La condición de posible accidente se tomó a partir de que 1 o más vehículos no se hayan movido de su posición tomando como punto de referencia las coordenadas de los mismos, si estos no presentan un desplazamiento por un tiempo determinado y se detecta presencia de personas, se considera como un accidente y se envía una alerta para que se almacene en la base de datos, la cual es tomada en cuenta para la toma de decisiones en el proyecto de semáforos inteligentes.

- ✓ Desarrollar un componente software basado en la Visión Artificial que permita determinar el congestionamiento vehicular en tiempo real

Este objetivo, al igual que el anterior se logró al delimitar la región de interés, descartando las zonas innecesarias y objetos erróneos, el clasificador en cascada desarrollado con imágenes tomadas desde la perspectiva de un semáforo (pasos peatonales ubicados a la altura de un semáforo), ayudó a que la detección de vehículos sea sobre el 90%, más de lo que se esperaba en el inicio del presente trabajo. La condición de posible congestionamiento se tomó a partir de que un número determinado de vehículos se presenten por un determinado tiempo, sin importar que el tráfico esté fluyendo. De igual

forma se almacena una alerta en la base de datos, lo cual ayuda en la toma de decisiones en el proyecto de semáforos inteligentes.

✓ Integrar los componentes de software de detección de accidentes y congestiónamiento vehicular a través de una interfaz gráfica de usuario.

Finalmente, luego de lograr desarrollar varios clasificadores, con los cuales se logró desarrollar los algoritmos con las condiciones para determinar un posible accidente o congestiónamiento, se procedió a implementar una interfaz gráfica de usuario, la cual permitió presentar los resultados obtenidos, cabe destacar que la interfaz gráfica de usuario no será tomada en cuenta en el proyecto de semáforos inteligentes, ya que lo que es necesario son los algoritmos desarrollados en C++.

h. Conclusiones

Luego de haber concluido con el desarrollo del presente proyecto de fin de carrera, es necesario dar a conocer algunas conclusiones, las mismas que a continuación se detallan:

- Los clasificadores son muy ventajosos con respecto a las técnicas tradicionales de visión artificial, ya que el margen de error es más reducido y el tiempo de respuesta de los algoritmos son muy cortos al permitir analizar cada frame de un video en tiempo real, lo cual permite enviar alertas de posibles sucesos de manera inmediata. Para que los tiempos de respuesta de los algoritmos sean óptimos se requiere de una máquina con buenas prestaciones, mínimo que el procesador sea Intel core i5 con 4GB de memoria RAM.
- Al delimitar el área de interés se eliminan espacios innecesarios a ser analizados por los algoritmos, de esta manera se reduce el consumo de la memoria operativa, lo cual conlleva a una buena optimización de los recursos del sistema.
- Los algoritmos de detección de accidentes y congestionamientos, son un punto de partida para poder implementar un sistema de control total de los semáforos dentro de la ciudad, con lo cual se dotaría de inteligencia artificial a los mismos, ayudando al mejoramiento y ordenamiento del tránsito vehicular dentro de la ciudad.
- Se requiere tomar las suficientes muestras o imágenes positivas, para poder desarrollar un correcto entrenamiento de los clasificadores, a mayor número de imágenes positivas, mayor precisión en la detección de los objetos a reconocer.
- La visión artificial es un campo que aún no se ha explotado en nuestro medio, por lo cual es una gran oportunidad de llevar a cabo proyectos para solventar las necesidades tecnológicas de la ciudad, el presente trabajo es un punto de partida para futuros trabajos de visión artificial en materia de tránsito y mejoramiento del mismo.

i. Recomendaciones

Además de las conclusiones planteadas anteriormente, también es necesario realizar algunas recomendaciones a considerar al momento de desarrollar un Sistema basado en Visión Artificial. A continuación se detallan estas recomendaciones:

- La selección de la cámara es muy importante, ya que se necesita tener imágenes de muy buena calidad para poder desarrollar un buen entrenamiento en los clasificadores, una cámara de 5MP sería suficiente tomando en cuenta que la cámara IP a utilizar en las pruebas es de 1MP. Además tener en cuenta la iluminación, ya que una de las grandes desventajas de la visión artificial es que en condiciones de poca luz no detecta el 100% de los objetos.
- Antes de tomar la decisión de qué técnicas y algoritmos de visión artificial utilizar, se debe hacer un análisis de lo que se pretende desarrollar, si se requiere que el tiempo de respuesta sea rápido, lo mejor es utilizar herramientas como Qt que trabaja en una forma eficiente con el lenguaje C++ y se integra fácilmente con la librería de Visión Artificial OpenCv.
- Utilizar un número de imágenes positivas con respecto a las negativas en una relación de 2 a 1, por ejemplo si se utilizan 1000 imágenes positivas, se deben utilizar 500 negativas. Además si se utiliza el algoritmo de Viola-Jones tener en cuenta la escala y las dimensiones del objeto a detectar, por ejemplo si se va a detectar un vehículo utilizar una escala $m \times m$, pero si se va a detectar una persona, utilizar una escala $m \times n$, de esta manera se puede utilizar varias escalas en la codificación de los algoritmos lo cual permite disminuir los tiempos de respuesta significativamente, por ende el rendimiento de la aplicación es mucho mejor.
- Utilizar una metodología de desarrollo ágil, lo cual permita enfocarse rápidamente en los requerimientos y desarrollo del sistema. De igual manera utilizar herramientas libres, evitando gastos innecesarios. Además se puede obtener fácilmente soporte técnico en los foros de discusión, tal es el caso de Ubuntu, Qt y OpenCv, las cuales se utilizaron en el presente trabajo, logrando los resultados de forma satisfactoria.

Trabajos futuros

El desarrollo de este tipo de sistemas da lugar a nuevas líneas de investigación, sean por ejemplo la mejora del mismo o sirviendo de base para el desarrollo de otros Sistemas mucho más robustos. Siendo algunos de los trabajos futuros los siguientes:

- Desarrollo de un sistema basado en Visión Artificial para detectar invasión de carriles.
- Desarrollo de un sistema basado en Visión Artificial para detectar invasión del paso cebra cuando el semáforo está en rojo.
- Sistemas basados en Visión Artificial para la detección de congestionamientos y accidentes de tránsito, utilizando sensores de presencia y sensores de impacto, los cuales vienen integrados en los vehículos modernos.
- Aplicaciones móviles, ya sea en Android o iOS, basados en Visión Artificial para la detección de señales de tránsito y cambio de estado en un semáforo.
- Sistema de reconocimiento de peatones para evitar atropellamientos.
- Seguimiento de objetos a través de un drone, por ejemplo vehículos reportados como robados.

j. Bibliografía

- [1] UGR, «La inteligencia artificial,» Universidad de Granada, 31 05 2007. [En línea]. Available: http://www.ugr.es/~setchift/docs/conciencia_capitulo_2.pdf. [Último acceso: 01 05 2015].
- [2] Nebrija, «Nebrija,» Introducción a la Inteligencia Artificial, 2010. [En línea]. Available: http://www.nebrija.es/~cmalagon/ia/transparencias/introduccion_IA.pdf. [Último acceso: 20 04 2015].
- [3] C. Platero, «Introducción a la Visión Artificial,» Universidad Politécnica de Madrid, Madrid, 2009.
- [4] Y. Gonzalez, «DMI,» [En línea]. Available: http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Aplicaciones_VC.PDF. [Último acceso: 10 05 2015].
- [5] Infaimon, «Infaimon,» [En línea]. Available: <http://blog.infaimon.com/2014/05/la-termografia-como-herramienta-de-diagnostico-en-medicina-deportiva/>. [Último acceso: 10 05 2015].
- [6] infaimon, «Infaimon,» [En línea]. Available: <http://blog.infaimon.com/2011/11/proyecto-magic-key-la-informatica-en-la-lucha-contra-la-info-exclusion/>. [Último acceso: 10 05 2015].
- [7] Infaimon, «Infaimon,» [En línea]. Available: <http://blog.infaimon.com/2014/05/analisis-de-imagen-en-el-estudio-inervacion-traqueal-anormal-en-ratas-con-hernia-diafragmatica-congenita-experimental/>. [Último acceso: 10 05 2015].
- [8] UDC, «sabia.tic.udc,» [En línea]. Available: <http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/aplicaciones.html>. [Último acceso: 10 05 2015].
- [9] dimages, «dimages,» 2011. [En línea]. Available: <http://www.dimages.es/Tutorial%20A./introduccion/resolucion.htm#top>. [Último acceso: 10 10 2014].
- [10] E. Anguiano, «Imagen digital,» Universidad Autónoma de Madrid, Madrid, 2010.

- [11] A. d. L. B. H. José Maximiliano Rivera de la Cruz, «Segmentación de imágenes de placas vehiculares usando técnica de crecimiento de regiones,» Escuela Superior Politécnica del Litoral, Guayaquil, 2011.
- [12] E. d. I. F. y. F.M.Trespaderne, «Imágenes Binarias,» Librovision, 2011. [En línea]. Available: <http://www.librovision.eii.uva.es/pdf/cap4.pdf>. [Último acceso: 16 05 2015].
- [13] K. J. A., «Base para la vision artificial,» Universidad Abierta Interamericana, 2011. [En línea]. Available: <http://imgbiblio.vaneduc.edu.ar/fulltext/files/TC099930.pdf>. [Último acceso: 10 05 2015].
- [14] «Tipos de imágenes y formatos - APRENDE TIC,» [En línea]. Available: <https://sites.google.com/site/ticvalcarcel/optimizacion-de-imagenes-para-internet/tipos-de-imagenes-y-formatos>. [Último acceso: 21 10 2014].
- [15] «docs,» [En línea]. Available: <http://docs.gimp.org/es/plugin-in-sobel.html>. [Último acceso: 04 05 2015].
- [16] J. V. Rebaza, «Detección de bordes mediante el algoritmo de Canny,» Universidad Nacional de Trujillo, Perú, 2010.
- [17] A. Martínez, «Solveet,» [En línea]. Available: <http://www.solveet.com/exercises/Detector-de-Bordes-Sencillo/150>. [Último acceso: 04 05 2015].
- [18] T. y. A. Departamento de Ingeniería en Electrónica, «Detección de bordes en una imagen,» Universidad de Jaén, 2006. [En línea]. Available: http://www4.ujaen.es/~satorres/practicas/practica3_vc.pdf. [Último acceso: 10 05 2015].
- [19] A. d. L. B. H. José Maximiliano Rivera de la Cruz, «Segmentación de imágenes de placas vehiculares usando técnica de crecimiento de regiones,» Escuela Superior Politécnica del Litoral, Guayaquil, 2011.
- [20] J. G. C. Alma E. Martínez Licon, «Definición de una red neuronal para clasificación por medio de un programa evolutivo,» Universidad Autónoma Metropolitana, Iztapalapa, 2001.
- [21] «Redes neuronales,» [En línea]. Available: <http://avellano.fis.usal.es/~lalonso/RNA/>. [Último acceso: 04 05 2015].
- [22] D. R. M., «Extracción automática de caras en imágenes captadas con móviles Android,» 2012. [En línea]. Available:

<http://upcommons.upc.edu/TT/bitstream/2099.1/15485/1/78399.pdf>. [Último acceso: 10 05 2015].

- [23] A. B. Dragolici, «Detección, localización e identificación biométrica de caras en imágenes, Métodos y evaluación en el marco NIST-MBGG,» Universidad Autónoma de Madrid, 06 2010. [En línea]. Available: http://atvs.ii.uam.es/files/2010_0602AndreiDragolici. [Último acceso: 10 05 2015].
- [24] P. V. a. M. Jones, «Robust Real-Time Face Detection,» *International Journal of Computer Vision*, vol. 57, 2004.
- [25] Wikipedia, «Wikipedia.org,» [En línea]. Available: http://es.wikipedia.org/wiki/C%C3%A1mara_digital. [Último acceso: 10 04 2015].
- [26] Wikipedia, «Wikipedia.org,» [En línea]. Available: http://es.wikipedia.org/wiki/C%C3%A1mara_IP. [Último acceso: 10 04 2015].
- [27] UBUNTUMX, «ubuntumx,» [En línea]. Available: <http://www.ubuntumx.org/queesubuntu.php>. [Último acceso: 18 04 2015].
- [28] WIKIPEDIA, «wikipedia,» [En línea]. Available: <http://es.wikipedia.org/wiki/Ubuntu>. [Último acceso: 18 04 2015].
- [29] WIKIPEDIA, «wikipedia,» [En línea]. Available: [http://es.wikipedia.org/wiki/Qt_\(biblioteca\)](http://es.wikipedia.org/wiki/Qt_(biblioteca)). [Último acceso: 18 04 2015].
- [30] J. C. O. Pérez, «Multi-clasificación discriminativa por partes mediante Códigos Correctores de Errores,» 29 06 2012. [En línea]. Available: Facultad de Matemáticas Universidad de Barcelona. [Último acceso: 10 04 2015].
- [31] WIKIPEDIA, «wikipedia,» [En línea]. Available: <http://es.wikipedia.org/wiki/OpenCV>. [Último acceso: 18 04 2015].
- [32] OPENCV, «opencv,» [En línea]. Available: <http://opencv.org>. [Último acceso: 18 04 2015].
- [33] ELVEX, «elvex,» [En línea]. Available: <http://elvex.ugr.es/idbis/db/docs/lifecycle.pdf>. [Último acceso: 18 04 2015].
- [34] «EcuRed,» [En línea]. Available: http://www.ecured.cu/index.php/Visi%C3%B3n_Artificial. [Último acceso: viernes abril 2014].

- [35] E. G. Santillán, Mayo 2008. [En línea]. [Último acceso: 25 Abril 2014].
- [36] K. J. A., «Base para la visión artificial,» Universidad Abierta Interamericana, 2011. [En línea]. Available: <http://imgbiblio.vaneduc.edu.ar/fulltext/files/TC099930.pdf>. [Último acceso: 10 05 2015].
- [37] D. R. M., «Extracción automática de caras en imágenes captadas con móviles Android,» 2012. [En línea]. Available: <http://upcommons.upc.edu/TT/bitstream/2099.1/15485/1/78399.pdf>. [Último acceso: 10 05 2015].
- [38] etitudela, «etitudela,» [En línea]. Available: www.etitudela.com/celula/downloads/visionartificial.pdf. [Último acceso: 24 04 2014].
- [39] arteuna, «arteuna,» [En línea]. Available: <http://www.arteuna.com/talleres/lab/ediciones/libreria/Virilio-Maquinadelavision.pdf>. [Último acceso: 24 04 2014].
- [40] J. B. C. M., «Diario Centinela,» [En línea]. Available: <http://www.diariocentinela.com.ec/transito-vehicular>. [Último acceso: 24 04 2014].
- [41] I. L. Espejo, «ilopez,» [En línea]. Available: <http://www.ilopez.es/proyectos/fisicayelectronica/SemPLC.pdf>. [Último acceso: 23 04 2014].
- [42] E. Comercio, «elcomercio,» [En línea]. Available: http://www.elcomercio.ec/pais/Congestion-vehicular-ciudades-Ecuador_0_292770763.html. [Último acceso: 24 04 2014].
- [43] R. d. UTPL, «dspace.utpl,» [En línea]. Available: <http://dspace.utpl.edu.ec/bitstream/123456789/4021/1/JUAN>. [Último acceso: 24 04 2014].

k. Anexos

Anexo 1. Entrevista

**AGENCIA NACIONAL DE TRÁNSITO
UNIDAD DE CONTROL OPERATIVO DE TRÁNSITO LOJA
ENTREVISTA**

PARA: Dr. Pablo Ochoa

SECRETARIO ABOGADO

FECHA: 06-07-2015

Por medio del presente solicitamos nos ayude con la siguiente entrevista, ya que la información que pueda proporcionar servirá para el desarrollo del trabajo de titulación cuyo tema versa sobre el:” **DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES Y/O CONGESTIONAMIENTO VEHICULAR** “en la ciudad de Loja.

- 1. Indique los Lugares y las horas donde con frecuencia existen congestionamiento vehicular en la Ciudad.**

- 2. Indique los Lugares y las horas donde con frecuencia existen accidentes de tránsito en la Ciudad.**

- 3. De las siguientes opciones indique que vehículos producen más congestionamiento siendo 1 el mayor y 8 el menor.**

Buses	[]
Taxis	[]

Taxis Piratas	<input type="checkbox"/>
Carros Particulares	<input type="checkbox"/>
Ambulancias	<input type="checkbox"/>
Bomberos	<input type="checkbox"/>
Camionetas de carga	<input type="checkbox"/>

4. De las siguientes opciones indique que vehículos producen más accidentes de tránsito siendo 1 el mayor y 6 el menor.

Buses	<input type="checkbox"/>
Taxis	<input type="checkbox"/>
Taxis Piratas	<input type="checkbox"/>
Carros Particulares	<input type="checkbox"/>
Ambulancias	<input type="checkbox"/>
Camionetas de carga	<input type="checkbox"/>

5. Señale con una X las principales circunstancias o motivos por las cuales se producen congestionamiento vehicular en la Ciudad.

Vías estrechas	<input type="checkbox"/>
Desconocimiento de las leyes de tránsito.	<input type="checkbox"/>
Descuido por parte de los conductores.	<input type="checkbox"/>
Mala redistribución del tránsito.	<input type="checkbox"/>
Número de vehículos elevados para una ciudad pequeña.	<input type="checkbox"/>
Otros indique:.....	<input type="checkbox"/>

6. Elija las principales circunstancias o motivos por las cuales se producen accidentes de tránsito en la Ciudad.

Imprudencia por parte de los conductores	<input type="checkbox"/>
Imprudencia por parte de los peatones	<input type="checkbox"/>
Malas condiciones del vehículos	<input type="checkbox"/>
Manejar en estado etílico	<input type="checkbox"/>
Exceso de velocidad	<input type="checkbox"/>
Malas condiciones climáticas	<input type="checkbox"/>

7. De las siguientes opciones siendo 1 el mayor y 4 el menor indique que numero de vehículos están involucrados en los accidentes de tránsito de la Ciudad.

- | | |
|-------------|-----|
| Uno | [] |
| Dos | [] |
| Tres | [] |
| Más de tres | [] |

8. De las siguientes opciones siendo 1 el mayor y 5 el menor, ¿en qué porcentaje se estiman los daños de los vehículos involucrados en un accidente de tránsito?

- | | |
|-------------|-----|
| 10 % | [] |
| 20% | [] |
| 30% | [] |
| 40% | [] |
| Más del 40% | [] |

9. De las siguientes opciones siendo 1 el mayor y 5 el menor, ¿Por lo general a que distancia se ubican los vehículos impactados luego de sufrir un accidente de tránsito?

- | | |
|--------------------|-----|
| 0 metros | [] |
| Entre 0 y 1 metros | [] |
| Entre 1 y 5 metros | [] |
| Más de 5 metros | [] |

10. De las siguientes opciones siendo 1 el mayor y 4 el menor. ¿qué número de vehículos se puede considerar para llegar a determinar que existe un congestionamiento en una avenida principal? (ej. Avda. Universitaria)

- | | |
|-------------------------|-----|
| Menos de 10 vehículos | [] |
| 10 vehículos | [] |
| Entre 10 y 20 vehículos | [] |
| Más de 20 vehículos | [] |

11. De las siguientes opciones siendo 1 el mayor y 4 el menor, de existir el número de vehículos a considerarse para que exista un congestionamiento, ¿Qué tiempo se estima que debe transcurrir para dar por sentado que se ha producido un congestionamiento?

30 segundos []

Un minuto []

Entre 1 y 5 minutos []

Más de 5 minutos []

Firma entrevistado(a)

CI:

Cargo:

Anexo 2. Instalación de OpenCV en Ubuntu 14.04

En este anexo se detalla los pasos a seguir para instalar correctamente OpenCV en Ubuntu 14.04, los cuales son los siguientes:

1. Remover ffmpeg

```
sudo apt-get remove ffmpeg x264 libx264-dev
```

2. Agregar repositorios

```
sudo add-apt-repository ppa:mc3man/trusty-media
```

3. Actualizar repositorios

```
sudo apt-get update
```

4. Instalar ffmpeg

```
sudo apt-get install ffmpeg x264 libx264-dev
```

5. Instalar librerías necesarias

```
sudo apt-get install build-essential checkinstall git cmake libfaac-dev libjack-jackd2-dev  
libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libsdl1.2-dev libtheora-dev  
libva-dev libvdpau-dev libvorbis-dev libx11-dev libxfixes-dev libxvidcore-dev texi2html yasm  
zlib1g-dev
```

6. Instalar librerías streamer

```
sudo apt-get install libgstreamer0.10-0 libgstreamer0.10-dev gstreamer0.10-tools  
gstreamer0.10-plugins-base libgstreamer-plugins-base0.10-dev gstreamer0.10-plugins-  
good gstreamer0.10-plugins-ugly gstreamer0.10-plugins-bad gstreamer0.10-ffmpeg
```

7. Actualizar repositorios

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

8. Instalar librerías de soporte para OpenCV

```
sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev libjasper-dev  
libopenexr-dev cmake python-dev python-numpy python-tk libtbb-dev libeigen3-dev yasm  
libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev  
libxvidcore-dev libx264-dev libqt4-dev libqt4-opengl-dev sphinx-common texlive-latex-extra  
libv4l-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev default-jdk ant  
libvtk5-qt4-dev
```

9. Configurar e instalar x264

```
cd /home/andres/Librerias/x264  
./configure --enable-shared  
make  
sudo make install
```

10. Configurar e instalar ffmpeg-2.4.3

```
cd ~  
cd /home/andres/Librerias/ffmpeg-2.4.3  
./configure --enable-shared --enable-libmp3lame --enable-gpl --enable-libfaac --enable-  
libvorbis --enable-pthreads --enable-libfaac --enable-libxvid --enable-x11grab --enable-  
libx264 --enable-libtheora --enable-libdc1394 --enable-nonfree --disable-stripping --enable-  
avfilter --enable-libopencore-amrnb --enable-libopencore-amrwb --enable-version3  
make  
sudo make install
```

11. Instalar libgtk2.0-0 y libjpeg62-dev

```
cd ~  
sudo apt-get install libgtk2.0-0 libgtk2.0-dev  
sudo apt-get install libjpeg62 libjpeg62-dev
```

12. Configurar e instalar v4l-utils-1.6.0

```
cd ~  
cd /home/andres/Librerias/v4l-utils-1.6.0  
./configure --enable-shared  
make
```

```
sudo make install
```

13. Configurar e instalar libcdio-0.93

```
cd ~
```

```
cd /home/andres/Librerias/libcdio-0.93
```

```
./configure --enable-shared
```

```
make
```

```
sudo make install
```

14. Configurar e instalar vcdimager-0.7.24

```
cd ~
```

```
cd /home/andres/Librerias/vcdimager-0.7.24
```

```
./configure --enable-shared
```

```
make
```

```
sudo make install
```

15. Configuara e instalar xine-lib-1.2.6

```
cd ~
```

```
cd /home/andres/Librerias/xine-lib-1.2.6
```

```
./configure --enable-shared
```

```
make
```

```
sudo make install
```

16. instalar qt SDK

```
cd ~
```

```
wget http://download.qt-project.org/official_releases/qt/5.3/5.3.0/qt-opensource-linux-x64-5.3.0.run
```

```
chmod +x qt-opensource-linux-x64-5.3.0.run
```

```
./qt-opensource-linux-x64-5.3.0.run
```

17. Configuracion e instalacion de Opencv-2.4.9

```
cd ~
```

```
cd /home/andres/opencv-2.4.9
```

```
mkdir build
```

```
cd build
```

```
cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON  
-D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D  
BUILD_EXAMPLES=ON -D OPENCV_BUILD_3RDPARTY_LIBS=ON -D  
WITH_FFMPEG=ON -D WITH_GTK=ON -D WITH_OPENEXR=ON -D WITH_OPENNI=ON  
-D WITH_PNG=ON -D WITH_TBB=ON -D WITH_XINE=ON -D WITH_GSTREAMER=ON -  
D WITH_QT=ON -D WITH_OPENGL=ON -D WITH_VTK=ON ..
```

```
make
```

```
sudo make install
```

18. Configurar archivo opencv.conf

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
```

Agregar y luego guardar:

```
/usr/local/lib
```

19. Configurar archivo bash.bashrc

```
sudo ldconfig
```

```
sudo gedit /etc/bash.bashrc
```

Agregar y luego guardar:

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
```

```
export PKG_CONFIG_PATH
```

20. Cerrar todo y reiniciar

Anexo 3. Clasificador en cascada

Se realizó la recolección de 3000 imágenes que contengan el objeto a reconocer en nuestro caso vehículos, las cuales las denominamos con el nombre de **positivas** y 6000 imágenes denominadas **negativas**, en las cuales no debe estar presente el objeto a reconocer o los que se utilizaron en las imágenes positivas, en nuestro caso vehículos. Además las imágenes positivas y las negativas deben estar numeradas desde el 1 hasta el número de imágenes utilizadas (positivas de 1 a 3000 y negativas 1 a 6000). Y por último todas las imágenes en formato de mapa de bits (bmp).

Para la construcción del clasificador en el Sistema Operativo Windows se utiliza una herramienta denominada **tools**, la cual posee archivos .bat y .exe que facilitan la construcción del clasificador.

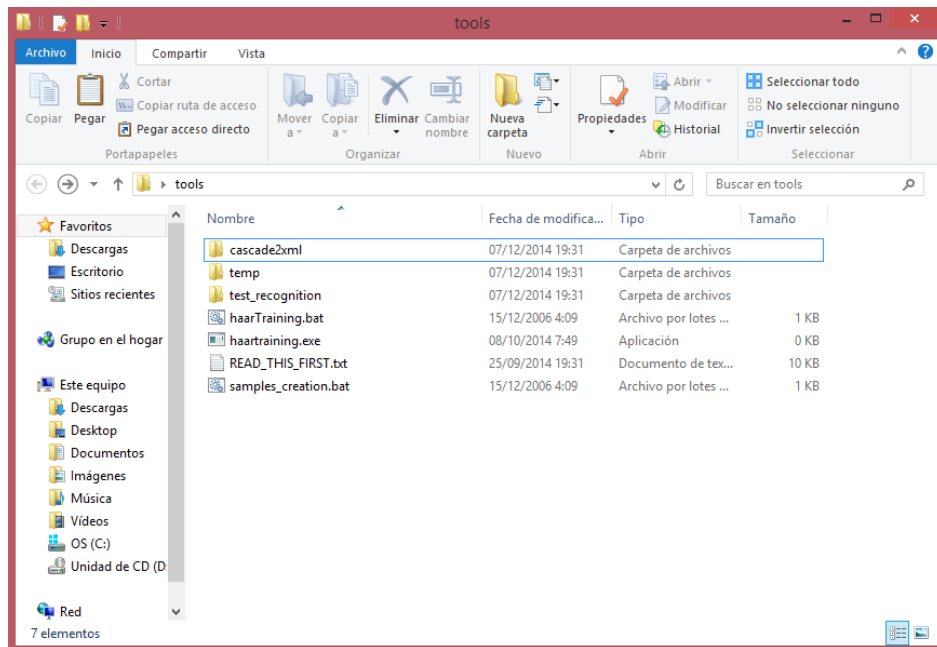


Fig. 1. Herramienta tools descomprimida.

Una vez recolectado las imágenes las ubicamos de manera correspondiente en los siguientes directorios **tools\temp\negative** para las negativas y **tools\temp\positive\rawdata** para las positivas.

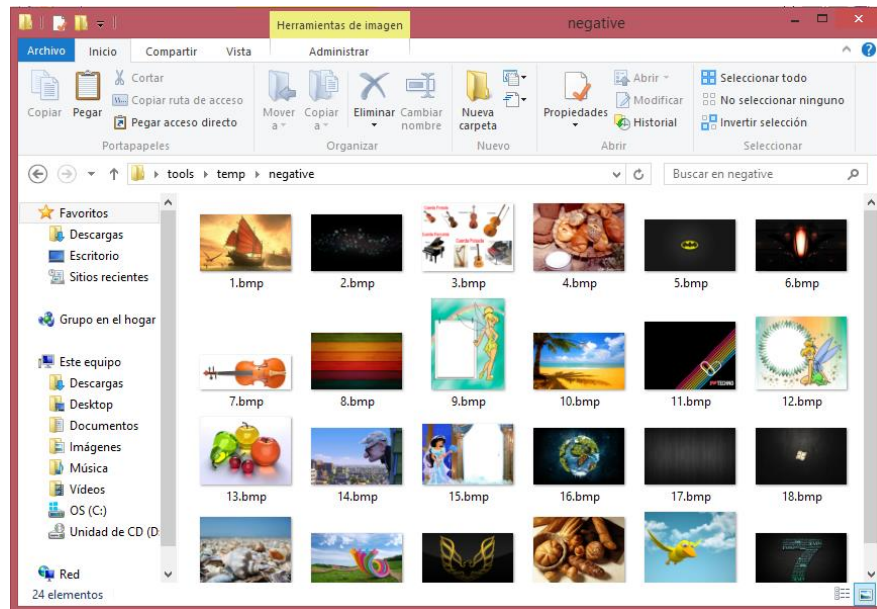


Fig. 2. Directorio de las imágenes negativas.

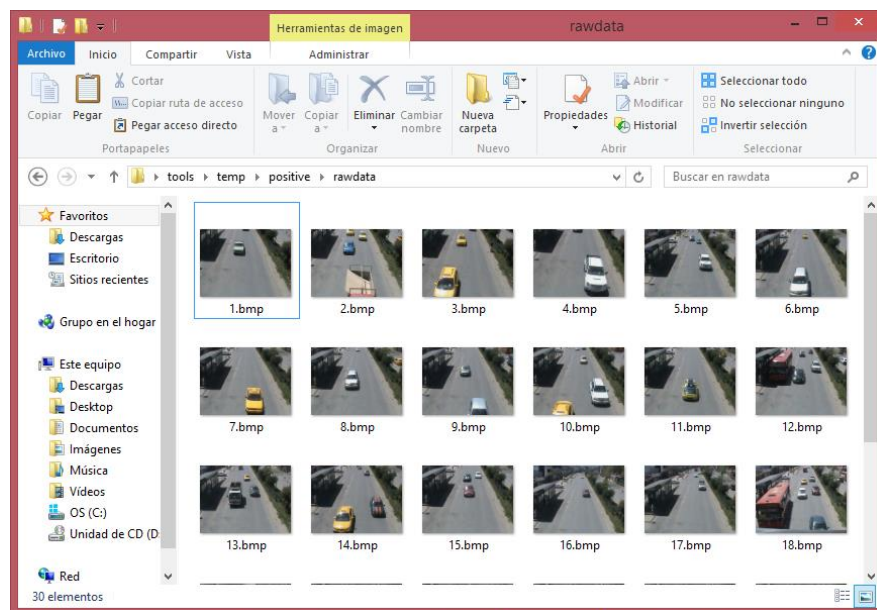


Fig. 3. Directorio de las imágenes positivas.

En el directorio donde se ubicaron las imágenes negativas, se encuentra el archivo **create_list.bat**, lo ejecutamos y generará un archivo **create_list.txt**

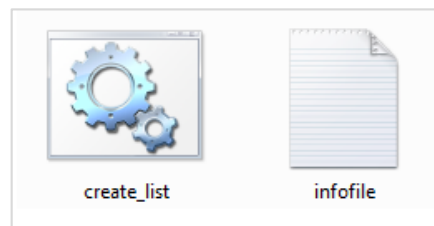


Fig. 4. Archivo .bat y .txt generado.

El archivo infofile.txt contiene el nombre de las 5000 imágenes negativas con su respectiva extensión.

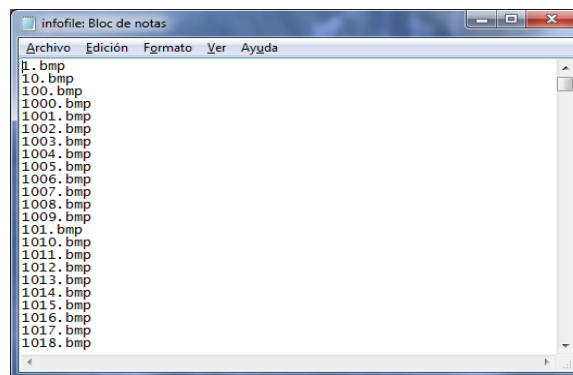


Fig. 5. Contenido del archivo infofile.txt

En el directorio de las imágenes positivas se encuentra el archivo **objectmarker.exe**, lo ejecutamos y presentará las siguientes interfaces. La interfaz de la parte superior indica la imagen que contiene el objeto a reconocer. En la interfaz de la parte inferior indica la información adicional de la imagen que se generará al señalar el objeto, el cual se encierra en un rectángulo de color fucsia (Seleccionar con el mouse, con la barra espaciadora para

generar los datos y presionar enter para pasar a la siguiente imagen). Se debe realizar todo este proceso para todas las imágenes positivas, es decir señalar el objeto en todas las imágenes.

La información adicional generada representa las coordenadas de la ubicación del objeto y las dimensiones de la parte seleccionada. Por ejemplo en la imagen 1.bmp al seleccionar el objeto, genera las coordenadas x=4 y=4 y las dimensiones 253x169.

Se debe realizar este proceso con todas las imágenes positivas, en este caso las 3000 imágenes. Cuando se finalice con todas las imágenes en el archivo info.txt se genera la información adicional especificada anteriormente.

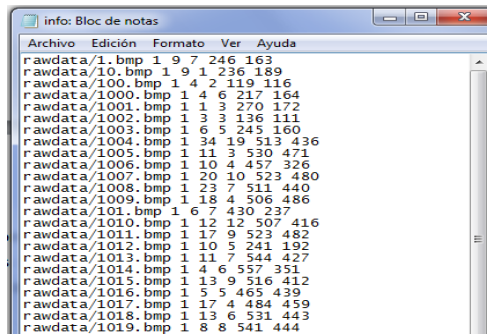


Fig. 6. Información adicional generada

Ahora procedemos a generar el archivo **.vec** que contendrá las muestras de cada una de las imágenes. En Windows debemos ejecutar lo siguiente:

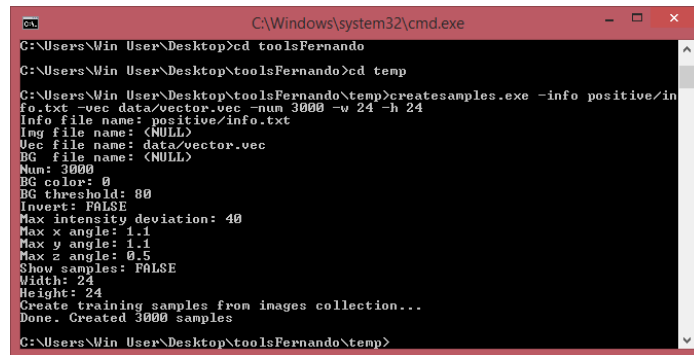
createsamples.exe -info positive/info.txt -vec data/vector.vec -num 3000 -w 24 -h 24

Donde:

- **createsamples:** es el nombre de la herramienta
- **info:** es la ubicación del archivo con el índice las imágenes positivas
- **vec:** es el nombre del archivo de salida con la muestra generada
- **num:** cantidad de imágenes positivas

- **w**: ancho de la muestra de salida
- **h**: alto de la muestra de salida.

Para ello abrimos la consola de Windows **cmd** y le damos la ruta de la carpeta **temp** ubicada en **tools** en este caso está en el escritorio:



```

C:\Windows\system32\cmd.exe
C:\Users\Min User\Desktop>cd toolsFernando
C:\Users\Min User\Desktop\toolsFernando>cd temp
C:\Users\Min User\Desktop\toolsFernando\temp>createsamples.exe -info positive/info.txt -vec data/vector.vec -num 3000 -w 24 -h 24
Info file name: positive/info.txt
Log file name: <NULL>
Vec file name: data/vector.vec
BG file name: <NULL>
Num: 3000
BG color: 0
BG threshold: 00
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 24
Height: 24
Create training samples from images collection...
Done. Created 3000 samples
C:\Users\Min User\Desktop\toolsFernando\temp>

```

Fig. 7. Creación del archivo **.vec**.

En el directorio **\temp\data** la carpeta **tools** se genera el archivo **vector.vec**

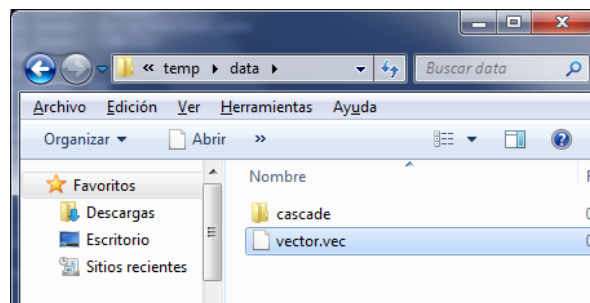


Fig. 8. Archivo **vector.vec** creado.

Finalmente procedemos a realizar el entrenamiento. Para este proceso nos valdremos de otra herramienta llamada **haartraining.exe** ubicado en el directorio **tools\temp** y el archivo índice generado con las imágenes negativas y el archivo de muestras generado en la etapa anterior. Ejecutamos lo siguiente:

haartraining.exe -data data/cascade -vec data/vector.vec -bg negative/infofile.txt -npos 3000 -nneg 6000 -nstages 20 -mem 1024 -mode ALL -w 24 -h 24 -nonsym

```

C:\Windows\system32\cmd.exe - haartraining.exe -data data/cascade -vec dat...
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Win User>cd Desktop
C:\Users\Win User\Desktop>cd toolsFernando
C:\Users\Win User\Desktop\toolsFernando>cd temp
C:\Users\Win User\Desktop\toolsFernando\temp>haartraining.exe -data data/cascade
-vec data/vector.vec -bg negative/infofile.txt -npos 3000 -nneg 6000 -nstages 20
-mem 1000 -mode ALL -w 24 -h 24 -nonsym
Data dir name: data/cascade
Vec file name: data/vector.vec
BG file name: negative/infofile.txt
Num pos: 3000
Num neg: 6000
Num stages: 20
Num splits: 1 (stump as weak classifier)
Mem: 1000 MB
Symmetric: FALSE
Min hit rate: 0.995000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: ALL
Width: 24
Height: 24
Max num of precalculated features: 19418
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost): misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 9.53674e-007

```

Fig. 9. Inicio del entrenamiento generando la primera tabla. Donde:

- **data data/cascade:** nombre y dirección que le damos al clasificador
- **vec data/vector.vec:** nombre y dirección del archivo .vec creado anteriormente
- **bg negative/infofile.txt:** lista de imágenes negativas
- **npos 3000:** número de imágenes positivas
- **nneg 6000:** número de imágenes negativas
- **nstages 20:** número de fases o estados (A más fases, mejor clasificador).
- **mem 1004:** espacio en memoria asignado para el entrenamiento
- **mode ALL:** método recomendado para el entrenamiento
- **w 24 -h 24:** tamaño exacto utilizado en la creación de las muestras
- **nonsym:** Simetría, si el objeto a buscar es simétrico verticalmente será un entrenamiento más rápido.

```

C:\Windows\system32\cmd.exe - haartraining.exe -data data/cascade -vec dat...
27! 72%|-1.430228| 0.995283| 0.545174| 0.095912|
28! 71%|-1.375867| 0.995283| 0.503266| 0.087567|
29! 71%|-1.371645| 0.995283| 0.482946| 0.081882|
Stage training time: 31220.48
Number of used features: 29
Parent node: 10
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
+-----+
| 0! 1! 2! 3! 4! 5! 6! 7! 8! 9! 10! 11! 12! 13! 14! 15! 16! 17! 18! 19 |
+-----+
| 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19 |
+-----+
Parent node: 19
*** 1 cluster ***
POS: 2743 3000 0.914333
NEG: 5486 7.17543e-007
BACKGROUND PROCESSING TIME: 201520.04
Required leaf false alarm rate achieved. Branch training terminated.
Total number of splits: 0
Tree Classifier
Stage
+-----+
| 0! 1! 2! 3! 4! 5! 6! 7! 8! 9! 10! 11! 12! 13! 14! 15! 16! 17! 18! 19 |
+-----+
| 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19 |
+-----+
Cascade performance
POS: 2743 3000 0.914333
_54%

```

Figura 10. Finalizando el entrenamiento generando la tabla 19.

Al finalizar el proceso de entrenamiento, en el directorio **cascade2xml\data** de la carpeta **tools**, se generan 20 carpetas numeradas desde 0 a 19, cada una contiene una fase o estado del proceso de entrenamiento representado en archivos **.txt**.

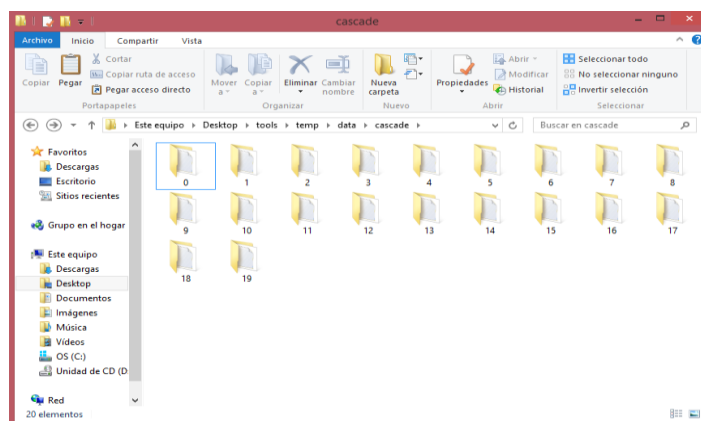


Fig. 11. Carpetas que contienen los estados generados.

Ahora generamos el archivo con extensión **.xml**, el cual será nuestro clasificador final para ello utilizaremos la herramienta **convert.bat** del directorio **tools\lcascade2xml**, lo

ejecutamos y se crea un archivo con el nombre **output.xml** que es el clasificador final que se utilizará para el reconocimiento de los vehículos.

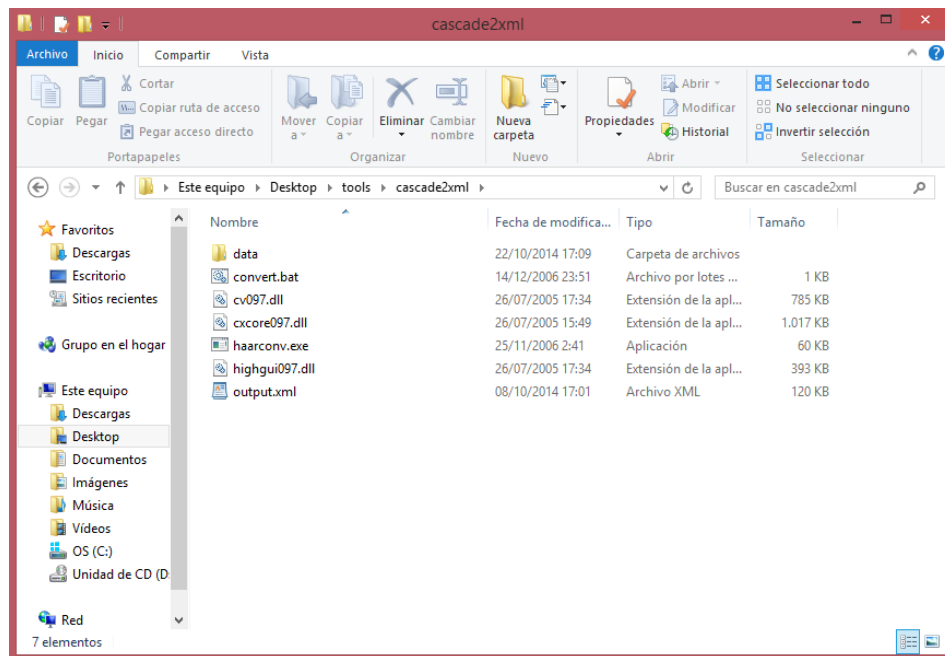


Fig. 12. Conversión a xml del archivo final (clasificador)

El proceso de entrenamiento dura varios días dependiendo del número de imágenes utilizadas, mientras más imágenes se utilicen mejor será el reconocimiento y la detección de los objetos. Este proceso duró alrededor de 3 días en un computador core i7 con memoria RAM 8 GB.

El entrenamiento, y por consiguiente el clasificador está estructurado en forma de árbol, para construir un clasificador robusto debería tener por lo menos dos nodos por fase (nsplits), pero a consecuencia de dicha estructura el tiempo de computación crecerá exponencialmente según avancen las fases. Parámetros muy altos harán bloquearse al PC, nunca debemos usar toda la memoria del ordenador. La cantidad de variables que existen independientes de estos parámetros (tamaño de las imágenes, procesador, tamaño del patrón, numero de imágenes positivas y negativas...) en este proceso hace imposible

hacerse una idea de lo que va a tardar, por eso es bueno empezar con valores modestos e ir incrementándolos, teniendo en cuenta que una buena estimación para que el clasificador sea robusto es más o menos una semana de entrenamiento, siendo menos importante cuales fueron los parámetros que se mejoraron. Podemos empezar por ejemplo con 2 splits, 0.998 de minhitrate, y 14 stages (muy orientativo).

No está preparado para procesadores de más de un núcleo, este problema hace que el entrenamiento vaya más rápido en una portátil, es más, mientras se ejecuta se puede comprobar en el administrador de tareas está usando exactamente un 25%.

Anexo 4. Artículo científico

Desarrollo de un sistema de Visión Artificial para la detección de accidentes y/o congestionamiento vehicular

Andrés Armijos, Pedro Aponte

Carrera de ingeniería de Sistemas, Universidad Nacional de Loja
Loja – Ecuador
adreami@gmail.com, fernandoaponte1103@gmail.com

Resumen. Este artículo es el resultado de la investigación exhaustiva de técnicas y algoritmos de procesamiento digital de imágenes, además del correspondiente trabajo de campo y diferentes herramientas para el diseño y programación. La visión artificial o visión por computador juega un papel muy importante para la consecución de los objetivos planteados, así mismo el avance tecnológico ha permitido que la misma permita en cierta forma recrear o simular las funciones del ojo humano.

Palabras Clave: Visión artificial, clasificador, vehículo.

Abstract. This article is the result of exhaustive research of techniques and algorithms of digital image processing, in addition to the corresponding fieldwork and different tools for design and programming. Computer vision or computer vision plays an important role in achieving the objectives, also the technological advancement has allowed it allows somehow recreate or simulate the functions of the human eye.

Keywords: Machine Vision, classifier, vehicle.

1 Introducción

El mundo actual en el que vivimos, está lleno de dispositivos capaces de realizar capturas de imágenes a grandes distancias, vigilar sectores específicos, cámaras de seguridad para grandes empresas, ojos de águila, etc. Todos estos dispositivos resultan de gran utilidad al momento de tomar una decisión por el ser humano, pero este no puede estar presente 24 horas, 365 días al año, es ahí donde entra la visión artificial, misma que juega un papel fundamental, haciendo que los dispositivos mencionados se conviertan en herramientas de capturas de imágenes pasivos a activos por si solos, capaces de tomar una decisión como lo haría un ser humano pensante e inteligente.

En la actualidad, es necesario determinar el tránsito vehicular de un determinado lugar donde con frecuencia existe congestionamiento de vehículos donde el tránsito es controlado por semáforos con temporizadores que se sitúan en intersecciones viales para regular el normal tráfico vehicular y peatonal, pero que resultan ineficientes al momento de la aglomeración vehicular, horas pico, provocando así una serie de problemas de contaminación ambiental accidentes de tránsito, contaminación por el ruido, etc..

De manera general en las grandes ciudades se encuentran los siguientes problemas:

- El control de tránsito se realiza mediante semáforos con temporizadores sincronizados.
- En horas pico se necesita la intervención un agente de tránsito para el control vehicular.
- El congestionamiento vehicular en las intersecciones en un intervalo de tiempo provocan posibles riesgos de colisiones.
- La aglomeración de vehículos provoca desesperación en los conductores y por ende tienden a utilizar en exceso las bocinas, provocando contaminación por ruido.

2 Herramientas utilizadas.

Al momento de desarrollar aplicaciones que estén basadas en Visión Artificial es muy importante saber elegir una librería como herramienta de desarrollo, aunque no sería muy recomendable utilizar solo una. Es probable que las deficiencias de una puedan ser resueltas por otra, o que para un problema concreto resulte aconsejable usar una librería específica como es el caso de Opencv.

A continuación se detallan algunas de las librerías más utilizadas en el desarrollo de aplicaciones que estén enfocadas en la Visión Artificial.

2.1 Ubuntu:



Fig. 1. Ubuntu. Última versión disponible 14.04 LTS

Ubuntu es una palabra Africana que significa 'Humanidad hacia otros', o 'Yo soy porque nosotros somos'. La distribución Ubuntu lleva el espíritu de Ubuntu al mundo del software.

Ubuntu es un sistema operativo desarrollado por la comunidad que es perfecto para laptops, computadoras de escritorio y servidores. Ya sea que lo utilices en el hogar, en la escuela o en el trabajo, Ubuntu contiene todas las aplicaciones que puedas necesitar, desde procesadores de texto y aplicaciones de email, hasta software para servidor web y herramientas de programación.

Ubuntu es y siempre será libre de costo. No pagas por una licencia de uso. Puedes descargar, usar y compartir Ubuntu con tus amigos, familiares, escuela o negocios libremente [6].

Su patrocinador, Canonical, es una compañía británica propiedad del empresario sudafricano Mark Shuttleworth. Ofrece el sistema de manera gratuita, y se financia por medio de servicios vinculados al sistema operativo y vendiendo soporte técnico. Además, al mantenerlo libre y gratuito, la empresa es capaz de aprovechar los desarrolladores de la comunidad para mejorar los componentes de su sistema operativo. Extraoficialmente, la comunidad de desarrolladores proporciona soporte para otras derivaciones de Ubuntu, con otros entornos gráficos, como Kubuntu, Xubuntu, Ubuntu MATE, Edubuntu, Ubuntu Studio, Mythbuntu, Ubuntu GNOME y Lubuntu [7].

2.2 Qt:



Fig 2. Qt. Última versión 4.8

Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores. Es desarrollada como un software libre y de código abierto a través

de Qt Project, donde participa tanto la comunidad, como desarrolladores de Nokia, Digia y otras empresas. Anteriormente, era desarrollado por la división de software de Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008. Qt es distribuida bajo los términos de GNU Lesser General Public License (y otras). Por otro lado, Digia está a cargo de las licencias comerciales de Qt desde marzo de 2011.

Qt es utilizada en KDE, entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings. También es usada en sistemas informáticos empotrados para automoción, aeronavegación y aparatos domésticos como frigoríficos.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales [8].

2.3 Opencv:



Fig 3. OpenCV. Última versión 3.0

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas [9].

OpenCV es liberado bajo una licencia BSD y por lo tanto es gratis, tanto para uso académico y comercial. Cuenta con interfaces de C++, C, Python y Java y es compatible con Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones de tiempo real. Escrito en optimizado C / C++, la biblioteca puede tomar ventaja de procesamiento multi-core. Habilitado con OpenCL, se puede aprovechar la aceleración de hardware de la plataforma de computación heterogénea subyacente. Adoptado en todo el mundo, OpenCV tiene más de 47 mil personas de la comunidad de usuarios y el número estimado de descargas superiores a 9 millones. Rangos de uso del arte interactivo, a la inspección de minas, mapas de costura en la web o a través de la robótica avanzada [10].

3 Metodología de desarrollo.

Para el desarrollo del proyecto se utilizó el modelo de ciclo de vida clásico, también denominado "modelo en cascada", se basa en intentar hacer las cosas bien desde el principio, de una vez y para siempre. Se pasa, en orden, de una etapa a la siguiente sólo tras finalizar con éxito las tareas de verificación y validación propias de la etapa. Si resulta necesario, únicamente se da marcha atrás hasta la fase inmediatamente anterior.

Este modelo tradicional de ciclo de vida exige una aproximación secuencial al proceso de desarrollo del software. Por desgracia, esta aproximación presenta una serie de graves inconvenientes, entre los que cabe destacar:

- Los proyectos reales raramente siguen el flujo secuencial de actividades que propone este modelo.
- Normalmente, es difícil para el cliente establecer explícitamente todos los requisitos al comienzo del proyecto (entre otras cosas, porque hasta que no vea evolucionar el proyecto no tendrá una idea clara de qué es lo que realmente quiere).
- No habrá disponible una versión operativa del sistema hasta llegar a las etapas [11].

4 Resultados.

4.1 Análisis de Requerimientos

En esta fase se realizó un análisis acerca de la perspectiva en la cual van a funcionar los algoritmos, en nuestro caso desde la perspectiva de un semáforo.

Para que la recolección de información se utilizó ciertas técnicas de recolección de datos, como son:

- La observación directa: que nos permitió conocer la forma en que transitan los vehículos y el área que debía ser limitada para el análisis vehicular.
- La entrevista: por parte del tutor de tesis que nos permitió conocer más a fondo la magnitud y el alcance del proyecto de semáforos inteligentes.

4.1.1 Requerimientos Funcionales

El Sistema permitirá:

TABLA 1. Requerimientos funcionales del sistema.

REF.	Descripción	Categoría	Técnicas de recolección
REF001	Captar imágenes y videos a través de una cámara.	Evidente	Observación
REF002	Captar videos previamente grabados.	Evidente	Observación
REF003	Detectar vehículos a través de un clasificador.	Evidente	Observación
REF004	Detectar personas a través de un clasificador.	Evidente	Observación
REF005	Delimitar el área en la cual se va a realizar la detección.	Evidente	Observación

REF006	Utilizar un temporizador para controlar los eventos que se den.	Evidente	Observación
REF007	Almacenar los eventos generados en una base de datos.	Oculto	Observación
REF008	Generar un reporte de los posibles congestionamientos generados.	Evidente	Entrevista
REF009	Generar un reporte de los posibles accidentes generados.	Evidente	Entrevista

4.1.2 Requerimientos no Funcionales

De la misma forma que los requerimientos funcionales, también existen los requerimientos no funcionales, en donde se dice que el sistema deberá:

TABLA 2. Requerimientos no funcionales del sistema.

REF.	Descripción	Categoría	Técnicas de recolección
REF001	Poseer una interfaz que sea amigable para el usuario.	Evidente	Entrevista
REF002	Facilitar su uso para poder realizar las pruebas apropiadas.	Evidente	Entrevista

4.2 Diseño

4.2.1 Algoritmo para la detección de vehículos y posible congestionamiento

A continuación se muestra el algoritmo que permite realizar la detección de los vehículos en tiempo real. El algoritmo está estructurado en un diagrama de flujo. Lo que hace el algoritmo al principio es una evaluación sobre si existe o no el objeto a detectar, en este caso sería el vehículo. Seguidamente se analiza si existe un determinado número de vehículos, en caso de ser afirmativo se inicia un contador por un determinado tiempo, si ambas condiciones se dan se presenta una alerta de posible congestionamiento. Todo esto se consigue con la ayuda de un clasificador, el cual se obtiene luego del entrenamiento que realiza este con varios modelos y formas de vehículos tomadas con una cámara digital desde la perspectiva de un semáforo. En la siguiente figura se muestra el esquema del algoritmo que permite realizar la detección de los vehículos y posible congestionamiento:

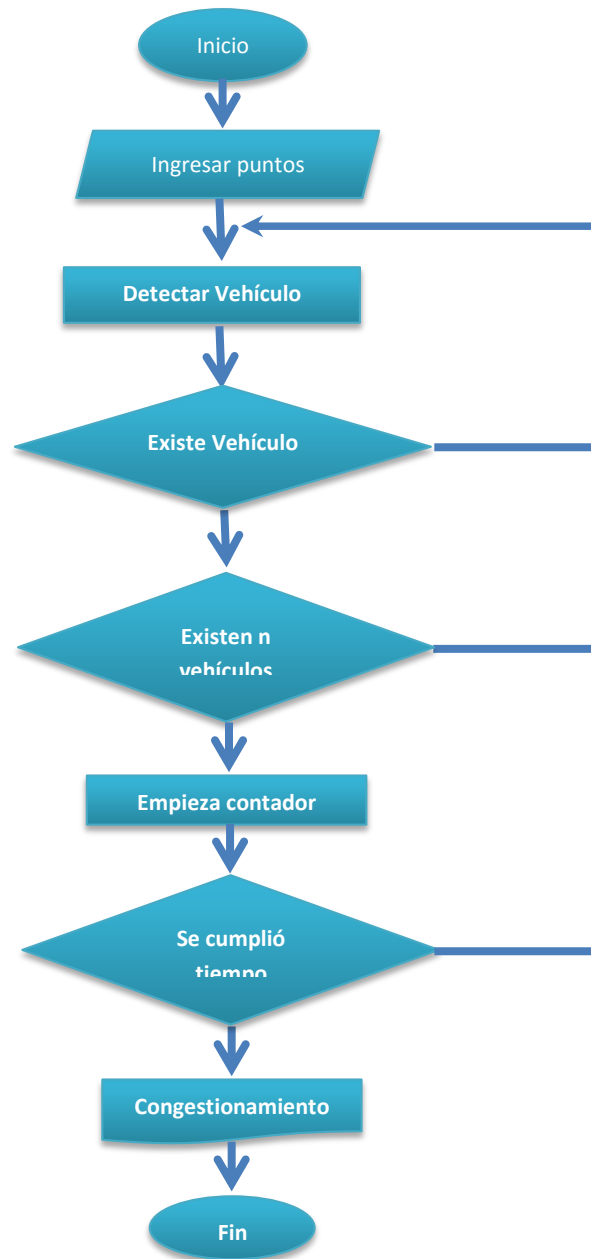


Diagrama 1. Diagrama de flujo del algoritmo para la detección de congestionamientos.

4.2.2 Algoritmo para la detección de vehículos y posible accidente

Este algoritmo funciona igual que el anterior. Lo que hace el algoritmo al principio es una evaluación sobre si existe o no un vehículo. Seguidamente se analiza si existe uno o más vehículos estáticos, en caso de ser afirmativo se inicia un contador por un determinado tiempo, si ambas condiciones se dan se presenta una alerta de posible accidente. En la siguiente figura se muestra el esquema del algoritmo que permite realizar la detección de los vehículos y posible accidente:

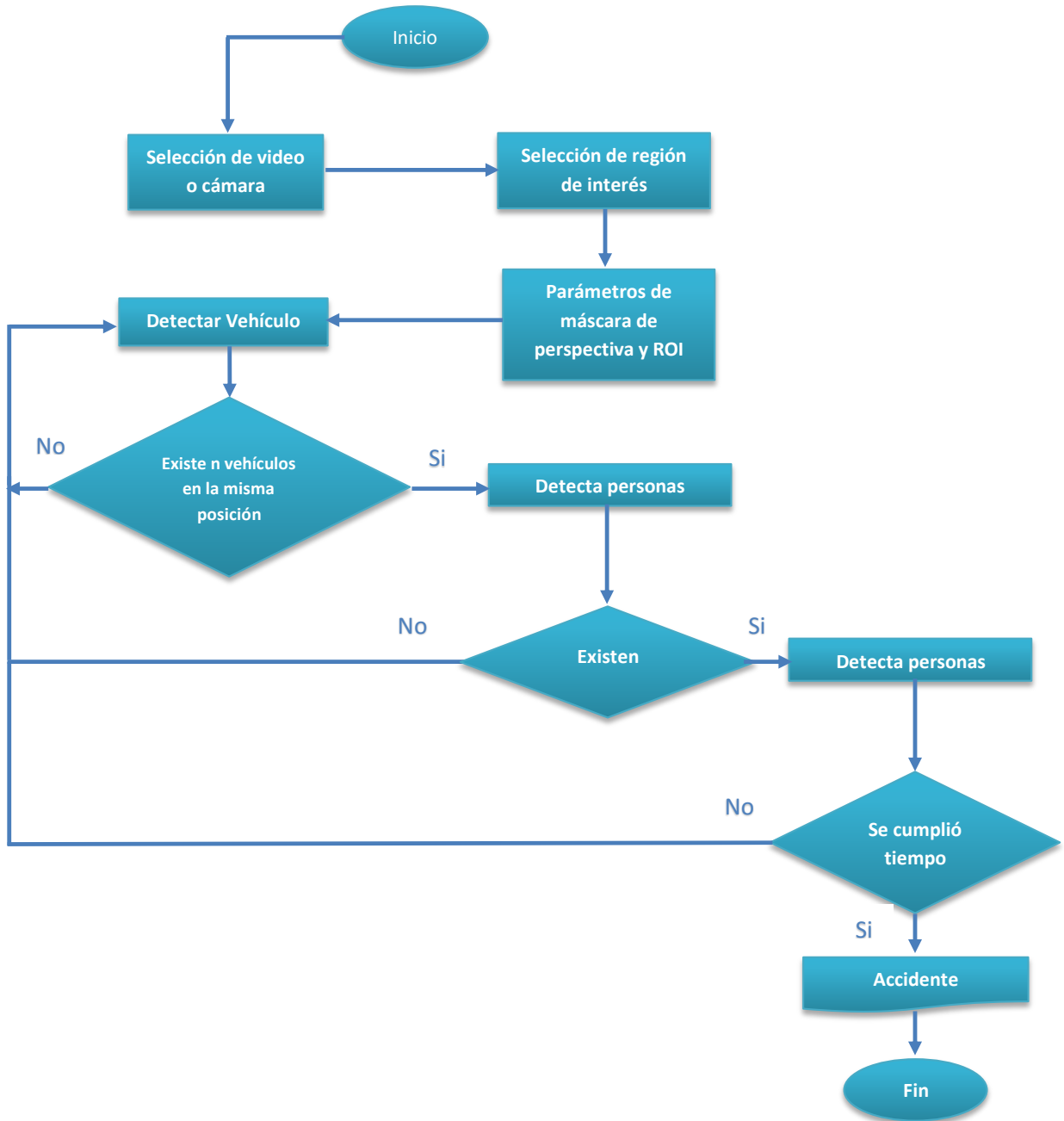


Diagrama 2. Diagrama de flujo del algoritmo para la detección de accidentes.

4.3 Diseño

4.3.1 Construcción del clasificador en cascada

Como se mencionó anteriormente para el desarrollo del algoritmo correspondiente a la detección de vehículos desde la perspectiva de un semáforo fue necesario construir un clasificador en cascada. Para realizar la detección de vehículos se construyó el clasificador en la plataforma Windows 8.1, debido a que se emplearon gran cantidad de imágenes (muestras).

4.3.2 Clasificador en cascada para la detección de vehículos

Se realizó la recolección de 3000 imágenes que contengan el objeto a reconocer en nuestro caso vehículos, las cuales las denominamos con el nombre de *positivas* y 6000 imágenes denominadas *negativas*, en las cuales no debe estar presente el objeto a reconocer o los que se utilizaron en las imágenes positivas, en nuestro caso vehículos. Además las imágenes positivas y las negativas deben estar numeradas desde el 1 hasta el número de imágenes utilizadas (positivas de 1 a 3000 y negativas 1 a 6000). Y por último todas las imágenes en formato de mapa de bits (bmp).

Para la construcción del clasificador en el Sistema Operativo Windows se utiliza una herramienta denominada *tools*, la cual posee archivos .bat y .exe que facilitan la construcción del clasificador.

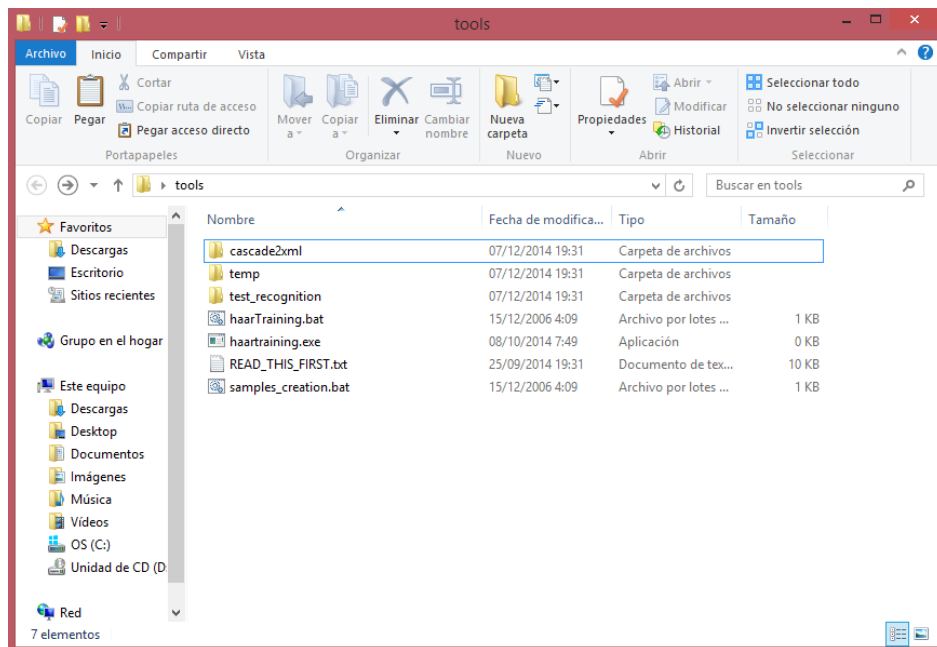


Fig. 4. Herramienta tools descomprimida.

Una vez recolectado las imágenes las ubicamos de manera correspondiente en los siguientes directorios *tools\temp\negative* para las negativas y *tools\temp\positive\rawdata* para las positivas.

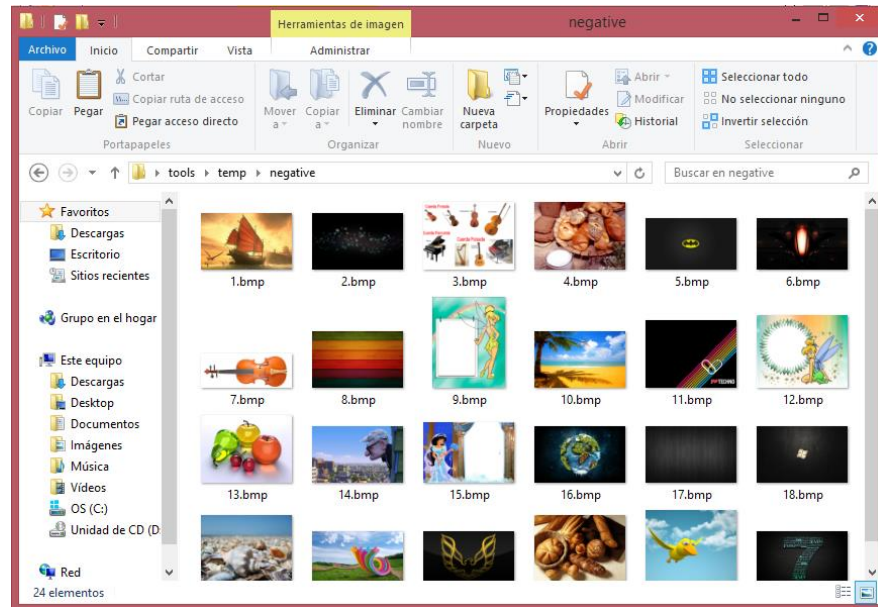


Fig. 5. Directorio de las imágenes negativas.

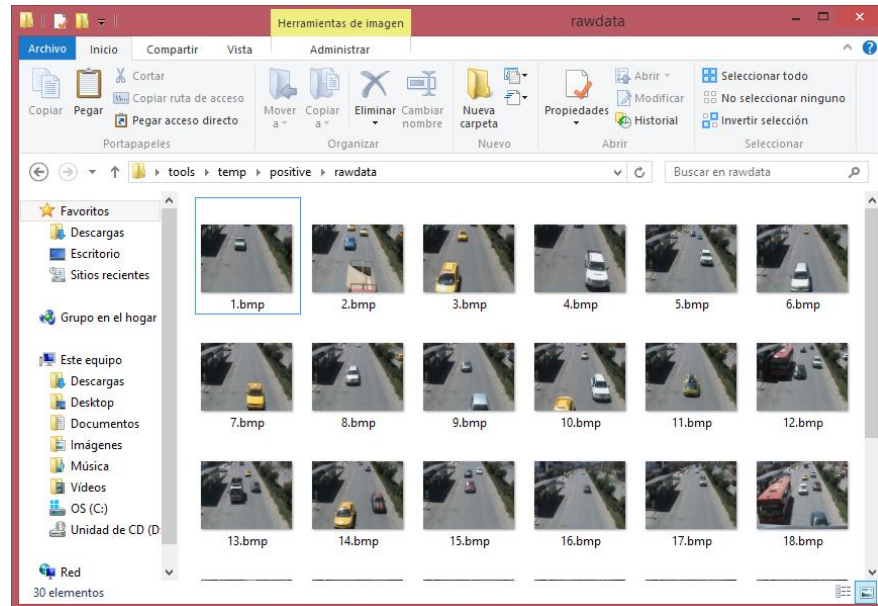


Fig. 6. Directorio de las imágenes positivas.

En el directorio donde se ubicaron las imágenes negativas, se encuentra el archivo **create_list.bat**, lo ejecutamos y generará un archivo **create_list.txt**

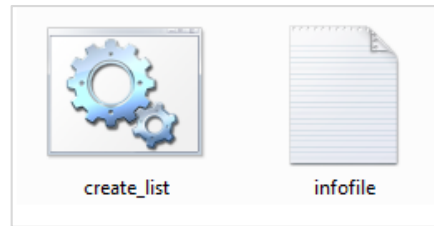


Fig. 7. Archivo .bat y .txt generado.

El archivo infolile.txt contiene el nombre de las 5000 imágenes negativas con su respectiva extensión.

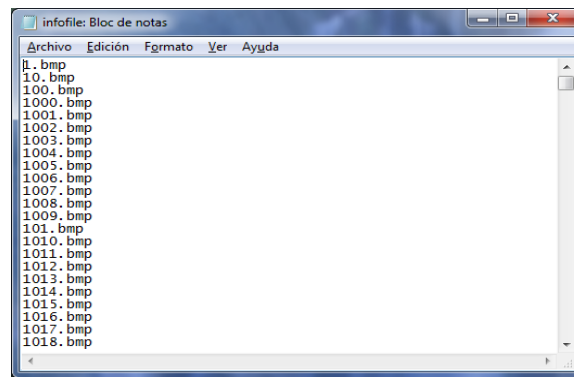


Fig. 8. Contenido del archivo infofile.txt

En el directorio de las imágenes positivas se encuentra el archivo **objectmarker.exe**, lo ejecutamos y presentará las siguientes interfaces. La interfaz de la parte superior indica la imagen que contiene el objeto a reconocer. En la interfaz de la parte inferior indica la información adicional de la imagen que se generará al señalar el objeto, el cual se encierra en un rectángulo de color fucsia (Seleccionar con el mouse, con la barra espaciadora para generar los datos y presionar enter para pasar a la siguiente imagen). Se debe realizar todo este proceso para todas las imágenes positivas, es decir señalar el objeto en todas las imágenes.

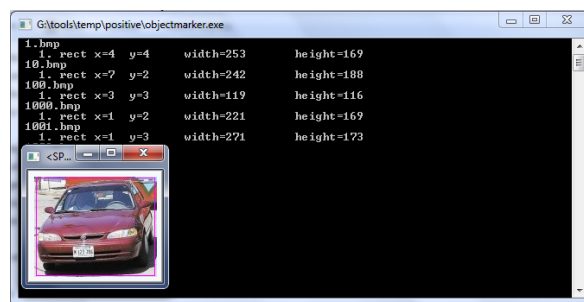


Fig. 9. Objeto seleccionado e información generada.

La información adicional generada representa las coordenadas de la ubicación del objeto y las dimensiones de la parte seleccionada. Por ejemplo en la imagen 1.bmp al seleccionar el objeto, genera las coordenadas x=4 y=4 y las dimensiones 253x169.

Se debe realizar este proceso con todas las imágenes positivas, en este caso las 3000 imágenes. Cuando se finalice con todas las imágenes en el archivo info.txt se genera la información adicional especificada anteriormente.

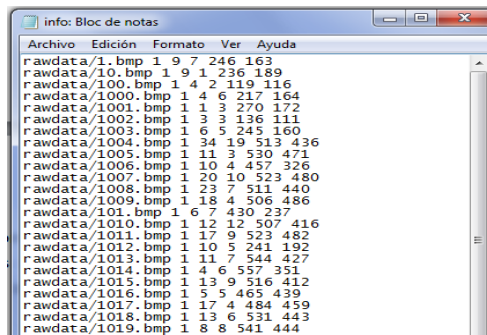


Fig. 10. Información adicional generada

Ahora procedemos a generar el archivo .vec que contendrá las muestras de cada una de las imágenes. En Windows debemos ejecutar lo siguiente:

createsamples.exe -info positive/info.txt -vec data/vector.vec -num 3000 -w 24 -h 24

Donde:

- **createsamples:** es el nombre de la herramienta
- **info:** es la ubicación del archivo con el índice las imágenes positivas
- **vec:** es el nombre del archivo de salida con la muestra generada
- **num:** cantidad de imágenes positivas
- **w:** ancho de la muestra de salida
- **h:** alto de la muestra de salida.

Para ello abrimos la consola de Windows **cmd** y le damos la ruta de la carpeta **temp** ubicada en **tools** en este caso está en el escritorio:

```

C:\Windows\system32\cmd.exe
C:\Users\Win User\Desktop>cd toolsFernando
C:\Users\Win User\Desktop\toolsFernando>cd temp
C:\Users\Win User\Desktop\toolsFernando\temp>createsamples.exe -info positive/info.txt -vec data/vector.vec -num 3000 -w 24 -h 24
Info file name: positive/info.txt
Img file name: <NULL>
Vec file name: data/vector.vec
BG file name: <NULL>
Num: 3000
BG color: 0
BG threshold: 00
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 24
Height: 24
Create training samples from images collection...
Done. Created 3000 samples
C:\Users\Win User\Desktop\toolsFernando\temp>

```

Fig. 11. Creación del archivo .vec.

En el directorio `\temp\data` la carpeta `tools` se genera el archivo `vector.vec`

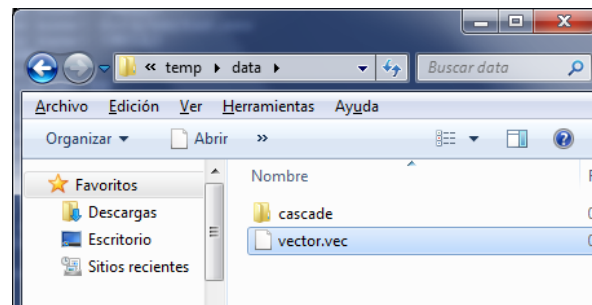


Fig. 12. Archivo `vector.vec` creado.

Finalmente procedemos a realizar el entrenamiento. Para este proceso nos valdremos de otra herramienta llamada `haartraining.exe` ubicado en el directorio `tools\temp` y el archivo índice generado con las imágenes negativas y el archivo de muestras generado en la etapa anterior. Ejecutamos lo siguiente:

haartraining.exe -data data/cascade -vec data/vector.vec -bg negative/infofile.txt -npos 3000 -nneg 6000 -nstages 20 -mem 1024 -mode ALL -w 24 -h 24 -nonsym

```

C:\Windows\system32\cmd.exe - haartraining.exe -data data/cascade -vec dat...
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Win User>cd Desktop
C:\Users\Win User\Desktop>cd toolsFernando
C:\Users\Win User\Desktop\toolsFernando>cd temp
C:\Users\Win User\Desktop\toolsFernando\temp>haartraining.exe -data data/cascade
-vec data/vector.vec -bg negative/infofile.txt -npos 3000 -nneg 6000 -nstages 20
-mem 1000 -mode ALL -w 24 -h 24 -nonsym
Data dir name: data/cascade
Vec file name: data/vector.vec
BG file name: negative/infofile.txt
Num pos: 3000
Num neg: 6000
Num stages: 20
Num splits: 1 (stump as weak classifier)
Mem: 1000 MB
Symmetric: FALSE
Min hit rate: 0.995000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: ALL
Width: 24
Height: 24
Max num of precalculated features: 19418
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost): misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 9.53674e-007

```

Fig. 13. Inicio del entrenamiento generando la primera tabla. Donde:

- **data data/cascade:** nombre y dirección que le damos al clasificador
- **vec data/vector.vec:** nombre y dirección del archivo .vec creado anteriormente
- **bg negative/infofile.txt:** lista de imágenes negativas
- **npos 3000:** número de imágenes positivas
- **nneg 6000:** número de imágenes negativas
- **nstages 20:** número de fases o estados (A más fases, mejor clasificador).
- **mem 1004:** espacio en memoria asignado para el entrenamiento
- **mode ALL:** método recomendado para el entrenamiento
- **w 24 -h 24:** tamaño exacto utilizado en la creación de las muestras
- **nonsym:** Simetría, si el objeto a buscar es simétrico verticalmente será un entrenamiento más rápido.

```

C:\Windows\system32\cmd.exe - haartaining.exe -data data/cascade -vec dat...
| 27| 72x|-1.438228| 0.995283| 0.545174| 0.095912|
| 28| 71x|-1.375867| 0.995283| 0.503266| 0.007567|
| 29| 71x|-1.371645| 0.995283| 0.482946| 0.001882|
Stage training time: 31220.48
Number of used features: 29
Parent node: 18
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
+-----+
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19|
+-----+
| 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19|
+-----+
Parent node: 19
*** 1 cluster ***
POS: 2743 3000 0.914333
NEG: 5486 7.17543e-007
BACKGROUND PROCESSING TIME: 201520.04
Required leaf false alarm rate achieved. Branch training terminated.
Total number of splits: 0
Tree Classifier
Stage
+-----+
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19|
+-----+
| 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19|
+-----+
Cascade performance
POS: 2743 3000 0.914333
_5.4%

```

Figura 14. Finalizando el entrenamiento generando la tabla 19.

Al finalizar el proceso de entrenamiento, en el directorio **cascade2xml\data** de la carpeta **tools**, se generan 20 carpetas numeradas desde 0 a 19, cada una contiene una fase o estado del proceso de entrenamiento representado en archivos **.txt**.

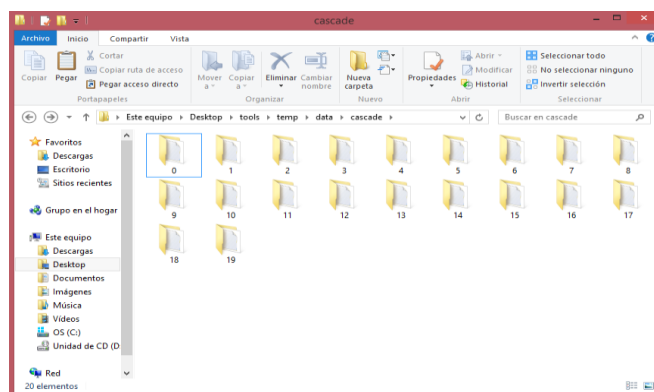


Fig. 15. Carpetas que contienen los estados generados.

Ahora generamos el archivo con extensión **.xml**, el cual será nuestro clasificador final para ello utilizaremos la herramienta **convert.bat** del directorio **tools\cascade2xml**, lo ejecutamos y se crea un archivo con el nombre **output.xml** que es el clasificador final que se utilizará para el reconocimiento de los vehículos.

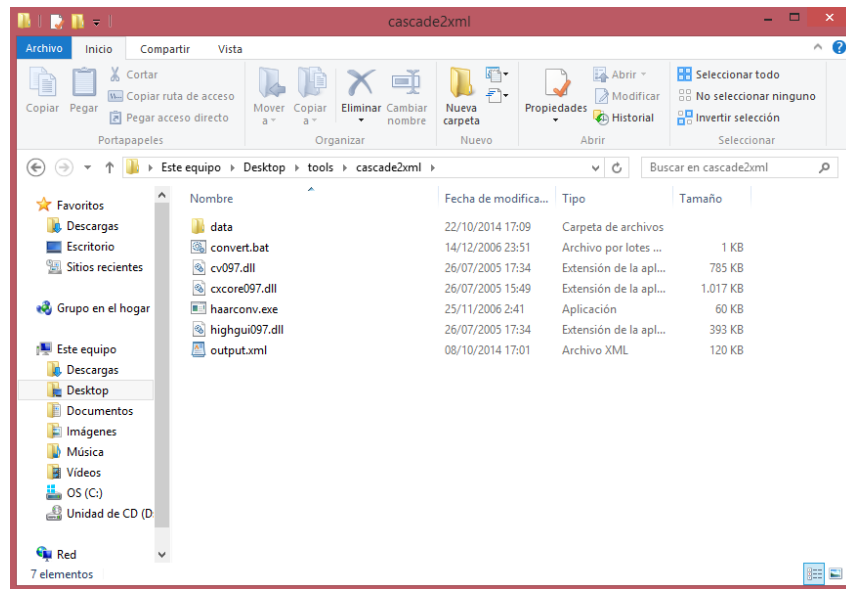


Fig. 16. Conversión a xml del archivo final (clasificador)

El proceso de entrenamiento dura varios días dependiendo del número de imágenes utilizadas, mientras más imágenes se utilicen mejor será el reconocimiento y la detección de los objetos. Este proceso duró alrededor de 3 días en un computador core i7 con memoria RAM 8 GB.

El entrenamiento, y por consiguiente el clasificador está estructurado en forma de árbol, para construir un clasificador robusto debería tener por lo menos dos nodos por fase (nsplits), pero a consecuencia de dicha estructura el tiempo de computación crecerá exponencialmente según avancen las fases. Parámetros muy altos harán bloquearse al PC, nunca debemos usar toda la memoria del ordenador. La cantidad de variables que existen independientes de estos parámetros (tamaño de las imágenes, procesador, tamaño del patrón, numero de imágenes positivas y negativas...) en este proceso hace imposible hacerse una idea de lo que va a tardar, por eso es bueno empezar con valores modestos e ir incrementándolos, teniendo en cuenta que una buena estimación para que el clasificador sea robusto es más o menos una semana de entrenamiento, siendo menos importante cuales fueron los parámetros que se mejoraron. Podemos empezar por ejemplo con 2 splits, 0.998 de minhitrate, y 14 stages (muy orientativo).

No está preparado para procesadores de más de un núcleo, este problema hace que el entrenamiento vaya más rápido en una portátil, es más, mientras se ejecuta se puede comprobar en el administrador de tareas está usando exactamente un 25%.

4.3.3 Implementación de los algoritmos

En la implementación de los algoritmos y la codificación del Sistema se utilizó el sistema Operativo Ubuntu 14.04, el Lenguaje de programación C++, el IDE QT Creator y SQLite para la Base de Datos.

4.3.4 Codificación del algoritmo para la detección de vehículos, posible congestionamiento y posible accidente.

Los algoritmos fueron programados en el lenguaje de programación C++, con la ayuda del framework Qt y la librería OpenCV.

4.4 Pruebas

Luego de haber construido e implementado el clasificador en los algoritmos correspondientes, pasamos a comprobar la precisión de los mismos.

4.4.1 Detección de vehículos

Para la construcción del clasificador en cascada para la detección de vehículos se emplearon 3000 imágenes positivas y 6000 imágenes negativas, obteniendo resultados muy efectivos, se obtuvieron los siguientes resultados:

Balance de detección con imágenes

TABLA 3. Balance de detección con imágenes vehículos.

Condición del día	%Aciertos	%F.P.	%F.N.
Lluvia	87,6	5,6	6,8
Llovizna con sol	95,4	2,3	2,3
Sol	96,2	1,9	1,9
Sombra	90,7	6,8	2,5
Promedio	92,5	4,1	3,4

TABLA 5. Balance de detección con imágenes personas.

Condición del día	%Aciertos	%F.P.	%F.N.
Sombreado	90	4	6
Normal	95	5	0
Variable	94	4	2
Promedio	93	4,3	2,7

TABLA 6. Balance de detección con imágenes personas y vehículos.

Condición del día	%Aciertos	%F.P.	%F.N.
Sombreado	100	0	0
Sombreado	100	0	0
Promedio	100	0	0

Como se puede observar el porcentaje de detección es por sobre el 90%.

En la siguiente figura se muestra la detección de vehículos usando el clasificador, que consistió en usar 3000 imágenes positivas y 6000 negativas.

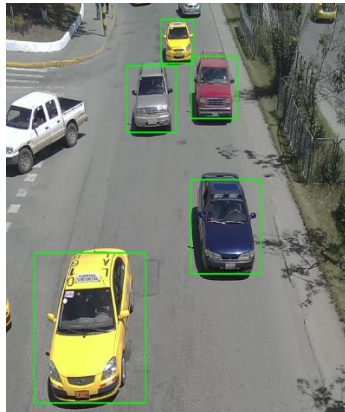


Fig. 17. Detección de vehículos utilizando el clasificador.

Balance de detección con video

TABLA 7. Balance de detección con videos, vehículos.

Lugar	%Aciertos	%F.P.	%F.N.
Terminal Terrestre	94	2	4
Terminal Terrestre	97,4	1,3	1,3
Calle Mercadillo	97,4	1,3	1,3
Promedio	96,3	1,5	2,2

TABLA 8. Balance de detección con videos, personas.

Lugar	%Aciertos	%F.P.	%F.N.
Calle Mercadillo	93,3	6,7	10
Alameda Real	100	0	10
Promedio	96,65	3,35	10

TABLA 9. Balance de detección con videos personas y vehículos.

Lugar	%Aciertos	%F.P.	%F.N.
Calle Mercadillo	100	0	20
Alameda Real	100	0	20
Promedio	100	0	20

En la mayoría de los balances o promedios de detección es sobre el 90%.

4.4.2 Registro del evento de posible congestionamiento

Para realizar las pruebas se tomaron como referencia el número de vehículos y el tiempo a considerar para que se produzca el evento. Se hicieron las pruebas en 2 videos, se tomó en cuenta que si existen 10 o más vehículos, se considere como un posible congestionamiento. El tiempo que se tomó para el temporizador fue de 60 segundos.

Cabe destacar que estos parámetros fueron tomados en base a las entrevistas realizadas.

Los resultados fueron los siguientes:

TABLA 10. Balance de detección con videos personas y vehículos.

Condición del día:		Soleado	
Lugar:	Paso peatonal Terminal Terrestre		
Hora:	01:00 P.M.		
	Número	%	
Veces que existieron más de 10 vehículos	2	100	
Posible congestionamiento detectado	2	100	
Falsos positivos	0	0	
Falsos negativos	0	0	



Diagrama 3. Prueba congestionamiento



Fig. 18. Detección de posible congestionamiento.

Sistema RCA

Reporte de posibles congestionamientos

Id	Tipo	Vehiculos	Fecha	Hora
1	Congestionamiento	10	23/07/15	23:18:26

Fig. 19. Registro del posible congestionamiento.

4.4.2 Registro del evento de posible accidente

Para realizar esta prueba se utilizó 2 videos en los cuales se detecta si dos o más vehículos no se han desplazado de su posición actual y si se detecta presencia de personas sea 1 o más. El tiempo que se tomó para el temporizador fue de 30 segundos.

Lo que hace el algoritmo es verificar si uno o más vehículos permanecen estáticos en las mismas coordenadas, tal como se puede observar en la figura anterior, si el vehículo que posee la coordenada (420, 415) y el vehículo que posee la coordenada (432,390), no se mueven de la misma coordenada por un lapso de 60 segundos y además se detecta la presencia de personas en el área delimitada, entonces se envía a guardar en la base de datos una alerta de un posible accidente.

Los resultados fueron los siguientes:

TABLA 11. Balance de detección con videos personas y vehículos.

Condición del día:	Sombreado	
Lugar:	Estacionamiento Alameda Real	
Hora:	05:30 P.M.	
	Número	%
Accidente presente	2	100
Posible accidente detectado	2	100
Falsos positivos	0	0
Falsos negativos	0	0



Diagrama 4. Prueba accidente



Fig. 20. Detección de posible accidente.

Sistema RCA

Reporte de posibles accidentes

Id	Tipo	Personas	Vehiculos	Fecha	Hora
42	Accidente	4	2	23/07/15	23:26:07

Fig. 21. Registro de posible accidente

5 Conclusiones.

- Los clasificadores son muy ventajosos con respecto a las técnicas tradicionales de visión artificial, ya que el margen de error es más reducido y el tiempo de respuesta de los algoritmos son muy cortos al permitir analizar cada frame de un video en tiempo real, lo cual permite enviar alertas de posibles sucesos de manera inmediata. Para que los tiempos de respuesta de los algoritmos sean óptimos se requiere de una máquina con buenas prestaciones, mínimo que el procesador sea Intel core i5 con 4GB de memoria RAM.
- Al delimitar el área de interés se eliminan espacios innecesarios a ser analizados por los algoritmos, de esta manera se reduce el consumo de la memoria operativa, lo cual conlleva a una buena optimización de los recursos del sistema.
- Los algoritmos de detección de accidentes y congestionamientos, son un punto de partida para poder implementar un sistema de control total de los semáforos dentro de la ciudad, con lo cual se dotaría de inteligencia artificial a los mismos, ayudando al mejoramiento y ordenamiento del tránsito vehicular dentro de la ciudad.
- Se requiere tomar las suficientes muestras o imágenes positivas, para poder desarrollar un correcto entrenamiento de los clasificadores, a mayor número de imágenes positivas, mayor precisión en la detección de los objetos a reconocer.
- La visión artificial es un campo que aún no se ha explotado en nuestro medio, por lo cual es una gran oportunidad de llevar a cabo proyectos para solventar las necesidades tecnológicas de la ciudad, el presente trabajo es un punto de partida para futuros trabajos de visión artificial en materia de tránsito y mejoramiento del mismo.

6 Referencias.

- [1] etitudela, «etitudela,» [En línea]. Available: www.etitudela.com/celula/downloads/visionartificial.pdf. [Último acceso: 24 04 2014].
- [2] arteuna, «arteuna,» [En línea]. Available: <http://www.arteuna.com/talleres/lab/ediciones/libreria/Virilio-Maquinadelavision.pdf>. [Último acceso: 24 04 2014].
- [3] J. B. C. M., «Diario Centinela,» [En línea]. Available: <http://www.diariocentinela.com.ec/transito-vehicular>. [Último acceso: 24 04 2014].
- [4] I. L. Espejo, «ilopez,» [En línea]. Available: <http://www.ilopez.es/proyectos/fisicayelectronica/SemPLC.pdf>. [Último acceso: 23 04 2014].
- [5] E. Comercio, «elcomercio,» [En línea]. Available: http://www.elcomercio.ec/pais/Congestion-vehicular-ciudades-Ecuador_0_292770763.html. [Último acceso: 24 04 2014].
- [6] UBUNTUMX, «ubuntumx,» [En línea]. Available: <http://www.ubuntumx.org/queesubuntu.php>. [Último acceso: 18 04 2015].
- [7] WIKIPEDIA, «wikipedia,» [En línea]. Available: <http://es.wikipedia.org/wiki/Ubuntu>. [Último acceso: 18 04 2015].
- [8] WIKIPEDIA, «wikipedia,» [En línea]. Available: [http://es.wikipedia.org/wiki/Qt_\(biblioteca\)](http://es.wikipedia.org/wiki/Qt_(biblioteca)). [Último acceso: 18 04 2015].
- [9] WIKIPEDIA, «wikipedia,» [En línea]. Available: <http://es.wikipedia.org/wiki/OpenCV>. [Último acceso: 18 04 2015].
- [10] OPENCV, «opencv,» [En línea]. Available: <http://opencv.org>. [Último acceso: 18 04 2015].
- [11] ELVEX, «elvex,» [En línea]. Available: <http://elvex.ugr.es/idbis/db/docs/lifecycle.pdf>. [Último acceso: 18 04 2015].
- [12] R. d. UTPL, «dspace.utpl,» [En línea]. Available: <http://dspace.utpl.edu.ec/bitstream/123456789/4021/1/JUAN>. [Último acceso: 24 04 2014].

Anexo 5. Certificado traducción de resumen



Prof. María Belén Novillo
DOCENTE DE FINE-TUNED ENGLISH

CERTIFICA:

Que el documento aquí compuesto es fiel traducción del idioma español al idioma inglés del resumen de la tesis titulada: "DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DE ACCIDENTES Y/O CONGESTIONAMIENTO VEHICULAR" de los autores: Andrés Armando Armijos Armijos y Pedro Fernando Aponte Rueda, egresados de la carrera de Ingeniería en Sistemas del Área de la Energía, las Industrias y los Recursos naturales No renovables de la Universidad Nacional de Loja.

Lo certifica en honor a la verdad y autoriza a los interesados hacer uso del presente en lo que a sus intereses convenga.

Loja, 26 de Junio de 2015

Prof. María Belén Novillo
DOCENTE DE FINE-TUNED ENGLISH



Anexo 6. Licencia Creative Commons



Desarrollo de un sistema de visión artificial para la detección de accidentes y/o congestionamiento vehicular. is licensed under a Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional License.